An Adaptable Variable Neighborhood Search for the Vehicle Routing Problem with Order Outsourcing

Huijink, S.; Kant, G.; Peeters, M.J.P.

Publication date:
2014

Document Version
Early version, also known as pre-print

No. 2014-062

AN ADAPTABLE VARIABLE NEIGHBORHOOD SEARCH FOR THE VEHICLE ROUTING PROBLEM WITH ORDER OUTSOURCING

By

Sybren Huijink, Goos Kant,
René Peeters

7 October, 2014

ISSN 0924-7815
ISSN 2213-9532

An Adaptable Variable Neighborhood Search for the Vehicle Routing Problem with Order Outsourcing

Sybren Huijink, Goos Kant & René Peeters

CentER and Department Econometrics and Operations Research, School of Economics and Management, Tilburg University, PO Box 90153, 5000 LE Tilburg, The Netherlands,
s.huijink@uvt.nl, g.kant@uvt.nl, m.j.p.peeters@uvt.nl

Abstract

In practice, many package transportation companies lower their costs by hiring outside carriers to serve orders that cannot be served efficiently by their own trucks. The problem which takes the order outsource option into account is the Vehicle Routing Problem with Private Fleet and Common Carrier. In this variant of the Vehicle Rouing Problem, orders are either delivered by an outside carrier, the common carrier, which receives an order specific price for this or by the own fleet, the private fleet, such that the total costs, which is the sum of the outside carrier costs and the delivery costs of the own fleet, is minimized. This paper presents two Adaptable Variable Neighborhood Search heuristics which are highly competitive and several new test instances which are based upon efficiency instead of necessity.

Keywords: Routing, Logistics, Metaheuristics, Pricing, Variable Neighborhood Search

1 Introduction

For many less-than-truckload companies it holds that the pick-up points of their orders lie close to the depot, while the delivery points are scattered over a much larger area. This implies that the costs of delivering the orders is quite costly and companies cooperate to save costs. One way to cooperate is by forming an alliance. In the Netherlands, there are the alliances Transmission, Teamtrans, Network Benelux and Distri-XL, which have the following similarities: The total area is divided into regions, and each member is responsible for one region, the home region. The alliance agrees on a pricing for which each member is obliged to deliver the order when it lies in its home region. Furthermore, each member decides on its own which orders outside its home region are outsourced, which is often decided at the beginning of the evening. The reallocation of the outsourced orders to the depot of the others is done by night-transportation, which is planned by the alliance but paid by the members. In order to determine which orders should be outsourced, the delivery planning is often split into two parts, one for the orders which lie inside the home region and one for the

orders outside. The planning is split into two since new orders which lie in the home region can arrive and that often the orders in the outside region cannot be combined with the orders in the home region due to time restrictions.

In this paper, we focus on the problem of deciding which orders to outsource and which routes to drive. This problem is most commonly known as the Vehicle Routing Problem with Private Fleet and Common Carrier (VRPPC) and the first modelling is from Chu (2005), although the decision which orders to outsource was already researched by Hall and Racer (1995). Since then, several heuristics have been proposed (Chu, 2005; Bolduc et al., 2007, 2008; Côté and Potvin, 2009; Potvin and Naud, 2011; Kratica et al., 2012; Stenger et al., 2013a,b). Archetti et al. (2009) researched the equivalent Capacitated Profitable Routing Problem (CPTP) together with the Team Orienteering Problem (TOP) (Boussier et al., 2007; Archetti et al., 2007). The research culminated in the work of Vidal et al. (2014) who took a different solution approach on the VRPPC. Other related problems to the VRPPC are Hot Rolling Scheduling Problems (Tang and Wang, 2006; Tang et al., 2009) while the TOP is closely related to the CPTP and the Traveling Salesman Problem with Profits (Feillet et al., 2005). Another example where a company has to decide which orders to outsource is that of small package shippers (SPS) (Stenger et al., 2013b). These SPS use outsourcing to decrease the last mile costs, for example hire Binnenstadsservice to deliver the last-mile.

We present two heuristics inspired by the work of Archetti et al. (2009), Stenger et al. (2013b) and Côté and Potvin (2009). Both of our heuristics are an Adaptable Variable Neighborhood search (AVNS), one slow (AVNS-Slow) and the other faster (AVNS-Fast). These AVNS are the first to use several completely different shaking moves, whereas both Archetti et al. (2009) and Stenger et al. (2013b) use only one type of move. Several of the moves in our heuristics are new and a Tabu Search is used as a local search method. It turns out that both our AVNS are highly competitive on the instances in the literature, especially for larger instances. Moreover, we improved the majority of the best known solutions during our testing. Finally, since the pricing in the literature has certain characteristics, e.g., high outsource costs, we tested our heuristics on instances with the pricings of Huijink et al. (2014).

The remainder of this paper is structured as follows. A mathematical model of the VRPPC and some extensions are presented in Section 2. In Section 3, the heuristics are described and the computational results are reported in Section 4. Finally, the conclusions are drawn in Section 5.

2 Problem Definition

The VRPPC is formally defined as follows. Let $\{0, 1, ..., n\}$ be the set of orders where 0 stands for the depot and let $\Omega = \{1, ..., |\Omega|\}$ be the set of all feasible routes. Set $a_i(r)$ equal to one if route $r \in \Omega$ delivers order $i \in [n] = \{1, ..., n\}$ and zero otherwise, p_i is the pre-determined outsource price incurred when order i is not delivered by the own fleet, $c(r)$ is the total cost of route r and there are m identical trucks with capacity Q. The decision variables are x_r where $x_r = 1$

implies that route r is driven. This gives the following set-packing model:

$$\text{minimize} \sum_{r\in\Omega} c(r)x_r + \sum_{i=1}^{n} p_i\Big(1 - \sum_{r\in\Omega} a_i(r)x_r\Big) \tag{1}$$

$$\text{subject to} \sum_{r\in\Omega} a_i(r)x_r \leq 1 \qquad \forall i \in [n] \tag{2}$$

$$\sum_{r\in\Omega} x_r \leq m \tag{3}$$

$$x_r \in \{0,1\} \qquad \forall r \in \Omega \tag{4}$$

The restriction that at most M trucks are driven, (3), and the integer restrictions, (4) are the same as in the classical VRP. The first difference is in (2), which states that each order is delivered at most once instead of exactly once. The second difference is a consequence of the first and is in the objective function, (1). Where in the VRP each order has to be delivered, there is now the option to not deliver an order and incur the cost of outsourcing the order, i.e., a penalty. Hence, the objective function, (1), minimizes the sum of the fixed vehicle costs, routing costs and common carrier costs.

An equivalent problem of the VRPPC is the Capacitated Profitable Tour Problem (CPTP) introduced by Archetti et al. (2009). In the CPTP, each order has a prize z_i which is collected when this order is visited by a private fleet vehicle, while in the VRPPC an outcourse cost p_i is incurred when an order is assigned to the common carrier. This changes the objective function of the set-partitioning problem of the CPTP as follows, while the constraints remain the same:

$$\text{maximize} \sum_{i=1}^{n} z_i \sum_{r\in\Omega} a_i(r)x_r - \sum_{r\in\Omega} c(r)x_r \tag{5}$$

Originally the VRPPC was introduced with a heterogenous fleet (Chu, 2005), but this is omitted in the models for clarity. Other extensions are: the private fleet should serve a predetermined amount (or ratio of the total demand) (Tang and Wang, 2006; Stenger et al., 2013a); that each private fleet vehicle can be used at most $t_{\max}$ time units (Stenger et al., 2013a,b); there are multiple own depots, multiple common carriers each with its own area of service (Stenger et al., 2013b,a), non-linear costs and a capacity restriction for the common carrier (Stenger et al., 2013a). Our heuristics can be adapted to handle all these constraints.

3 Solution method

We present two new adaptable variable neighbourhood searches and test the Reseeding-Tabu Search (R-TS) of Huijink et al. (2014) on more instances. For convenience we present the R-TS again. First, we discuss the Tabu Search (TS), which is used in all the heuristics, followed by the R-TS and the two AVNS. In order to simplify the notation, a virtual vehicle, vehicle 0, is used. This virtual vehicle contains some of the outsourced orders similarly as in Bolduc et al. (2008) and Stenger et al. (2013a,b). It has unlimited capacity, the costs will be

the outsource costs of the orders in the vehicle and the ordering, i.e., route, of these orders is not important. In Vidal et al. (2014), there is no virtual vehicle and all the orders are assigned to the available vehicles, but not all these orders are selected as being truly delivered. By q_i, we denote the capacity demand of order i and $d_{i,j}$ is the distance between order i and j where order 0 denotes the depot.

3.1 Tabu Search

A simple Tabu Search, which is inspired by Côté and Potvin (2009) is used as a local search method. Our TS will always return a feasible solution, even if the starting solution was infeasible and even while infeasible solutions are allowed during the search. An overview of our Tabu Search is presented in Algorithm 1 followed by an explanation of the parts used in the Tabu Search, i.e., making a solution feasible, the stopping criterion, the neighborhood, the best move, the execution of the move, updating of the infeasibility penalty, acceptance and the hierarchical objective function, and updating the tabu list.

Algorithm 1 Tabu Search (TS)

{Initialization phase}
Let s be the initial solution and set $s^{\text{global}} \leftarrow s$
Make s^{global} feasible
{Solution phase}
Set $n_{\text{iter}} = 0$ and $n_{\text{noimpr}} = 0$, $infeas = false$ and $n_{\text{inf}} = 0$
while $n_{\text{iter}} < N_{max}$ **and** $n_{\text{noimpr}} < N_{\text{noimpr}}$ **or** $n_{\text{iter}} < N_{\text{min}}$ **do**
  $n_{\text{iter}} = n_{\text{iter}} + 1$, $n_{\text{noimpr}} = n_{\text{noimpr}} + 1$ and $n_{\text{inf}} = n_{\text{inf}} + 1$
  {Search neighborhood}
  Search the neighborhood of s for the best move
  {Exection of the move}
  Execute the best move on s, 4*-Opt the two trucks that changed
  {Updating infeasibility penalty}
  if $\sum_{k \in T_{driven}(s)} I(k) > 0$ **then**
    $infeas = true$
  end if
  if $n_{\text{inf}} >= N_{\text{inf}}$ **then**
    if $infeas == true$ **then**
      $\lambda = \min\{\lambda_{max}, (1 + \lambda_{up})\lambda\}$
    else
      $\lambda = \max\{\lambda_{min}, \frac{\lambda}{(1+\lambda_{down})}\}$
    end if
    $infeas = false$ and $n_{\text{inf}} = 0$
    Recalculate solutions s and s^{global}
  end if
  {Acceptance}
  if $f(s) < f(s^{\text{global}})$ **and** s has no infeasibility in driven trucks **then**
    Set $s^{\text{global}} \leftarrow s$ and $n_{\text{noimpr}} = 0$
  end if
  {Updating the Tabu List}
  Update Tabu List
end while
Make s^{global} feasible
Return s^{global}

3.1.1 Making a solution feasible

A solution is made feasible similarly as is done by Archetti et al. (2007). Randomly a truck is selected that is infeasible, i.e., has more demand dan capacity, and randomly an order is removed from this truck. The removed order is feasibly inserted into a driven truck such that the total additional profit, which is price minus insertion distance, is maximized and strictly positive. If this is not

possible, i.e., the additional profit is negative or there is no feasible insertion, then it is tried to feasibly insert it into a non-driven truck, not truck 0, such that the profit is maximized and larger than zero. If there is still no feasible insertion with a strictly positive profit, the order is inserted into the virtual truck. This process is repeated until every truck is feasible. Ofcourse, truck 0 is always feasible.

3.1.2 Stopping criterion

The Tabu Search stops when either N_{noimpr} iterations have been executed without improvement or when the total number of iterations reaches N_{iter}, but it does not stop before $N_{\min}$ iterations have been executed.

3.1.3 Neighborhoods

The Tabu Search uses the union of the classical 1-shift move, which removes order i from its truck k and inserts it into truck k' in the best insertion place, and the 1-swap move, which removes orders i from its truck k and i' from truck k' and inserts i in k' and i' in k at the best insertion place. Note that $k \neq k'$.

3.1.4 Best Move

The best move is the move that yields the best solution, i.e., the lowest hierarchical f of Section 3.1.7, that is either not tabu, or it would yield a better solution than the best solution found so far, i.e., revoking of the tabu status.

3.1.5 Execution of the move

The best move is executed on the solution s and the two trucks that changed are improved by applying the 4*-*Opt* (Renaud et al., 1996) procedure. The 4*-*opt* uses a subset of 8 moves out of the available 48 4-*opt* moves. In 4-*opt*, the route is cut into 5 parts which are then glued back together in a different order. Furthermore, 4*-*opt* uses a test beforehand which determines whether or not a move has potential to improve the route. The 4*-*Opt* is never executed on truck 0.

3.1.6 Updating infeasibility penalty

At each iteration, it is checked whether the new solution has some infeasibility or not. If the new solution has any trucks, driven or not, that are not feasible, then this is memorized. After a number of iterations, which is determined by the parameter N_{inf}, the penalty for infeasibility λ is adapted as follows. If there was at least one infeasible solution, then λ is increased by a factor $1 + \lambda_{up}$, with the additional restriction that $\lambda \leq \lambda_{max}$. If there was no infeasible solution, then λ is decreased by a factor $1 + \lambda_{down}$, with the additional restriction that $\lambda \geq \lambda_{min}$. After the penalty has changed, the solutions s and s^{current} are re-evaluated.

3.1.7 Acceptance and hierarchical objective function f

A solution is accepted if it is better and if there is no infeasibility in the driven trucks, where the infeasibility is given by the square of the amount of capacity that does not fit in the truck and the infeasibility costs of a truck is the infeasibility of the truck multiplied by the infeasibility penalty λ. Since an improvement of the main objective, which are the total costs, is often the result of multiple iterations combined, we use a hierarchical objective function inspired by Archetti et al. (2009). In our function, we make a distinction between driven trucks and non-driven trucks, where a truck is driven if and only if the total operational costs of the truck, which is the sum of the fixed costs, the driving

costs of the route and the infeasibility costs, are less than the outsource costs of all the orders. Note that the costs of a truck is the minimum of the operational costs and the outsource costs and that this implies that the virtual truck is never driven.
The first level, $f_1(s)$, of the hierarchical function is the most important one and returns the total costs of the solution. The second objective, $f_2(s)$, steers the solution into areas where the amount of distance driven together with the total infeasibility is as small as possible. The third objective, $f_3(s)$, steers the solution into areas where new profitable routes are generated by simultaneously minimizing the outsource costs of the virtual truck and maximizing the outsource cost minus the distance of the other non-driven trucks. Note that a solution s is strictly better than s^*, which is denoted by $f(s) < f(s^*)$, if and only if $f_1(s) < f_1(s^*)$, or $f_1(s) = f_1(s^*)$ and $f_2(s) < f_2(s^*)$, or $f_1(s) = f_1(s^*)$, $f_2(s) = f_2(s^*)$ and $f_3(s) < f_3(s^*)$.

3.1.8 Updating the Tabu List

If an order i is removed from its truck k, then it is forbidden to re-insert order i in truck k for the next randomly chosen t iterations where $t \in [t_{lower}, t_{upper}]$. If an order is removed from a truck that is not driven, then it is also forbidden to reinsert the order into another non-driven truck for the same amount of time t.

3.2 Reseeding Tabu-Search

The Reseeding Tabu-Search and the initialization are from our previous paper (Huijink et al., 2014) and are presented again for completeness. Furthermore, we test this heuristic on instances which were not tested in our previous paper.

Algorithm 2 Reseeding Tabu (R-TS)

```
1: {Initialization phase}
2: Let k be a random integer between [0, m] and set s as the Profit Initialization with k
   trucks.
3: Set s^best ← s and set i = 1
4: repeat
5:    {Local Search}
6:    Execute Tabu Search on s
7:    {Acceptance}
8:    if f(s) < f(s^best) then
9:       Set s^best ← s
10:   else
11:      Set s ← s^best
12:   end if
13:   {Shaking}
14:   Execute Reseeding(0.5 + 0.1 · i)
15: until i>3
```

3.2.1 Initialization

In the VRPPC, orders have to be either driven or outsouced. In the literature (Côté and Potvin, 2009; Potvin and Naud, 2011; Stenger et al., 2013a,b), it is common to decide beforehand which orders to outsource. Furthermore, it is assumed that all the trucks are needed and a classical VRP initialization is used on the remaining orders. However, we do not assume that all the trucks are needed and opted for an initialization that makes a tradeoff depending on the costs of delivering the order with the own fleet and the outsource price instead of a decision beforehand (Huijink et al., 2014). In the first step of our

initialization, the number of trucks used and the seeds are chosen and in the second step the most profitable customers are inserted into the trucks. In the first step, we draw a random integer between 0 and m, where m is the number of available trucks. Then, the order which has the highest unit profit, i.e., $\frac{p_\ell - d_{0,\ell}}{q_\ell}$, is chosen as seed for the first free truck. In order to avoid setting seeds that are relatively close to this seed, the orders with the best unit profit compared to the seed, i.e., $\frac{p_\ell - d_{\text{seed},\ell}}{q_\ell}$ are temporarily inserted until the truck has reached half of its capacity. This procedure is repeated until the number of seeds equals the random integer chosen. In step two, the most profitable orders are inserted into the trucks which have a seed using a modification of the classical insertion heuristic. In this modification, the order that can be feasibly inserted and has the highest feasible unit insertion profit, i.e., $\frac{p_\ell - \text{InsertionDistance}}{q_\ell}$, is inserted into the corresponding truck if the insertion profit is larger than zero. This process is stopped when there are no feasible or profitable insertions left and all the remaining orders that are not yet inserted are outsourced, i.e., inserted into truck 0.

3.2.2 Local Search

The Reseeding Tabu-Seach uses the tabu search discussed in Algorithm 1 with $N_{\min} = 0$, $N_{\max} = 10000$, $N_{\text{noimpr}} = 5000$, $t_{lower} = 6$, $t_{upper} = 25$, $\lambda = 1$, $\lambda_{max} = 1000$, $\lambda_{min} = 0.001$, $\lambda_{down} = 1.9$, $\lambda_{up} = 1.1$ and $N_{infeas} = 4$.

3.2.3 Acceptance

The new solution is accepted if it is better than the old one.

3.2.4 Reseeding

The goal of reseeding is to go to another part of the search space without completely starting over. This is done by keeping a large part of the routes as seed and then apply the profit insertion.
First, take all the orders which are not driven, i.e., they are in trucks that are not driven, out of their trucks. For all the other trucks, keep a percentage, of the orders in the truck that has the highest ratio of penalties minus distance divided by the capacity. Keep this part into the truck and take all the other orders out of the truck. After all the trucks are adjusted, the insertion method used in the initialization is applied.

3.3 AVNS-F(ast)

The AVNS-Fast has as shaking moves several completely different moves as a local search method a Tabu Search. An overview of the heuristic is presented in Algorithm 3. The initialization of the AVNS-Fast is the same as in the Reseeding Tabu-Search, but is additionally followed by a short Tabu Search where $N_{\min} = 300$, $N_{\max} = 1000$, $N_{\text{noimpr}} = 75$, $t_{lower} = 6$, $t_{upper} = 25$, $\lambda = 1$, $\lambda_{max} = 1000$, $\lambda_{min} = 0.001$, $\lambda_{down} = 1.9$, $\lambda_{up} = 1.1$ and $N_{infeas} = 4$. The other steps, i.e., the stopping criterion, shaking moves and local search and finally the acceptance decision are discussed in their own sections.

3.3.1 Stopping criterion

The AVNS-Fast stops when either there has been no improvement of the best solution for $N_{\text{noimpr}} = 200$ iterations and the current best solution has not been improved for $N_{\text{jump}} = 2 \times M_{\max} = 40$ iterations, or when the total number of iterations reached $N_{\text{total}} = 1400$.

Algorithm 3 AVNS-Fast

{Initialization phase}
Generate initial solution s and improve s by using the Tabu Search heuristic
{Searching phase}
Set $s^{\text{global}} \leftarrow s$, $s^{\text{local}} \leftarrow s$ and the shaking moves $\mathcal{M} = \{0, ..., M_{\max} - 1\}$
Set $\kappa = 0$, $n_{\text{total}} = 0$, $n_{\text{noimpr}} = 0$ and $n_{\text{jump}} = 0$
while $n_{\text{total}} < N_{\text{total}}$ **and** ($n_{\text{noimpr}} < N_{\text{noimpr}}$ **or** $n_{\text{jump}} < N_{\text{jump}}$) **do**
  if $\kappa == 0$ **then**
    Randomly permutate $\mathcal{M}$
  end if
  {Shaking & Local Search}
  Execute move $M(\kappa)$ on s and improve s by using the Tabu Search heuristic
  {Acceptance}
  if $f(s) < f(s^{\text{local}})$ **then**
    if $f(s) < f(s^{\text{global}})$ **then**
      $s^{\text{global}} \leftarrow s$ and $n_{\text{noimpr}} = 0$
    end if
    $s^{\text{local}} \leftarrow s$ and $n_{\text{jump}} = 0$
  else
    if $f(s) > f(s^{\text{local}})$ **and** accept(n_{jump}) **then**
      $s^{\text{local}} \leftarrow s$ and $n_{\text{jump}} = 0$
    else
      $s \leftarrow s^{\text{local}}$
    end if
    $n_{\text{noimpr}} = n_{\text{noimpr}} + 1$ and $n_{\text{jump}} = n_{\text{jump}} + 1$
  end if
  $\kappa = (\kappa + 1) \mod M_{\max}$ and $n_{\text{total}} = n_{\text{total}} + 1$
end while
Return s^{global}

3.3.2 Shaking moves

Our AVNS-Fast uses the following shaking moves: Open-cyclic moves inspired by Stenger et al. (2013a), jumps inspired by Archetti et al. (2007), creation of a new route, destroying a route followed by the creation of a new route, dividing a route over two trucks, bomb moves which destroy an area and rebuild it, reseeding which takes orders out of trucks and rebuilds them, and extended Tabu Search. After many of these moves, we end up with trucks that are not full and need orders from other close driven trucks to become profitable. In order to shift orders between trucks, we created the shifting procedure. This procedure is discussed below together with the Shaking moves. After each shaking move, except the extended Tabu Searches, a Tabu Search is executed where $N_{\min}$ is a random integer between $[25, 45]$, $N_{\max}$ between $[300, 500]$ and N_{noimpr} between $[25, 35]$.

3.3.3 Shifting method

With some of the moves it is necessary that the trucks shift some orders to each other, e.g., when a new truck is created all the trucks need to make room for the new truck by providing some of the orders. In the shifting method, certain trucks, called the receiving trucks, receive orders from the providing trucks, where the providing trucks are not allowed to give away too many orders. However, after the shifting, the providing trucks need to be filled, therefore the procedure is repeated, each time with more restrictions on the amount of capacity that the providing trucks are allowed to provide. We choose five waves and in the first wave all feasible 1-*insert* moves from any providing truck, except moves from providing trucks which have less than *First* times their capacity and were driven at the start, to any receiving truck are consid-

Number	Move	#Trucks	#FirstOrders	#OtherOrders	Frequency
1	CyclicDriven	[2,2]	[3,4]	[0,0]	1
2	CyclicDriven	[2,2]	[2,3]	[2,3]	1
3	CyclicDriven	[2,2]	[3,4]	[3,4]	1
4	CyclicDriven	[2,4]	[2,4]	[2,4]	1
5	CyclicNotDriven	[2,2]	[5,6]	[2,3]	1
6	CyclicNotDriven	[2,2]	[2,3]	[2,3]	1
7	CyclicNotDriven	[2,4]	[2,4]	[2,4]	1
8	Create				2
9	Destroy				2
10	Split				1
11	Bomb	[3, m-1]	12.5%	40%	1
12	Bomb	[3, 6]	10%	40%	1
13	Jump		[3,8]		1
14	Jump		[5,12]		1
15	Reseeding	[2,6]	[5,11]		1
16	Reseeding	[5,m]	[6,11]		1
17	TabuSearch				2

Table 1: The moves of AVNS-Fast.

ered. Each of these 1-*insert* moves receives a score based on the unit insertion profit, i.e., $\frac{p_i - InsertionDistance_i}{q_i}$, which is either multiplied or divided by $(1 + DistanceNow_i - InsertionDistance_i)$, where $DistanceNow_i$ is the insertion distance of the order in the providing truck, if the providing truck drove at the start and the distance is less in the receiving truck or more, respectively. The order with the highest score is inserted into the corresponding receiving truck and the procedure is repeated until there are no more feasible or profitable insertions. In the second wave, the trucks which where providing and use at most a fraction *Second* of their capacity are the receiving trucks and the remaining providing trucks are again providing trucks. The insertion is repeated, but now the capacity restriction is *Second* for the providing driven trucks. The third wave is similar to the second, but now with *Third* as demand ratio. In the fourth wave all the driven trucks that where never receiving trucks are receiving trucks and all the non-driven trucks are providing trucks. In the fifth and final wave, the virtual truck is the providing truck and the other non-driven trucks are receiving trucks.

3.3.4 CyclicDriven and CyclicNotDriven

These moves are inspired by the ones of Stenger et al. (2013a) and cyclicly move a small number of orders from one truck to another truck. Our cyclic moves starts by selection a truck according to one of the selection methods. From this truck, a certain number of orders are removed according to another selection criterium. These orders are inserted into another driven truck that is not yet used in this cyclic move and that has the least insertion distance of these orders. This is repeated until the final truck has to be chosen. When choosing the final truck, the first truck is again allowed to be chosen. If the first truck is chosen, then the move is a closed cyclic move. After the cyclic move, the infeasibility of the trucks that are not driven is removed by randomly moving orders from the infeasible trucks to the non-driven trucks where it can be feasibly inserted and has the least additional distance. There are two types of moves, CyclicDriven, which tries to improve the solution by moving orders between driven trucks, and CyclicNotDriven, which tries to improve the solution by moving orders from non-driven trucks to driven trucks and vice versa. The characteristics of

the moves can be found in Table 1, where the number of trucks, orders from the first truck and from the other trucks are determined by a random integer in the bounds given. The CyclicDriven moves try to decrease the distance driven, hence we have the following four selection methods for the first truck, random, distance, unit distance and distance to other trucks.

1. *Random:* All driven trucks have the same probability of being selected.
2. *Distance:* The probability is proportional to the total distance traveled in this route. Hence, longer routes have higher probability of being selected.
3. *Unit Distance:* Similar to *Distance*, but now the distance traveled divided by the capacity used in the truck.
4. *Distance to Others:* The probability is proportional to the inverse of the least insertion cost of an order in this truck to other trucks. In this way, a truck is favored which has orders close to another truck.

To get promising moves, we have the following four selection methods for orders in the trucks, random, distance, average insertion distance and capacity.

1. *Random:* All orders have the same probability of being selected.
2. *Distance:* The probability is proportional to the sum of the the distances of the start and the end order of this sequence compared with the order before the first or the order after the last, respectively. Hence, sequences that are far away from the other orders in the route have a higher probability.
3. *Average Insertion Distance:* The probability is inversely proportional to the average insertion distance of the sequence in the closest non-chosen driven truck. By doing so, the orders that are close to another driven truck have a higher chance of being chosen.
4. *Capacity:* The probability is proportional to the total capacity of the sequence.

The next truck is the driven truck which is not yet chosen, barring the last choice where the first truck can again be chosen, and has the least insertion costs of the sequence of orders.

The CyclicNotDriven moves move orders from non-driven trucks to driven and vise versa. Therefore, we have the following selection methods for the first truck, which is always a non-driven truck, random, profit, unit profit and unit insertion profit in driven trucks.

1. *Random:* All non-driven trucks have the same probability of being selected.
2. *Profit:* The probability is proportional to $1+Profit$ if the Profit is positive and $\frac{-1}{-1+Profit}$ otherwise, where the Profit is the prices collected minus the distance driven. Furthermore, since truck 0 does not have a distance, it receives half the value of the truck with the lowest value. Hence, more profitable routes are more likely to be selected.

3. *Unit Profit:* The probability is proportional to $1 + ProfitOrder$ if ProfitOrder is positive and $\frac{-1}{-1+ProfitOrder}$ otherwise, where ProfitOrder is the price of the best order minus the distance of this order divided by its capacity. Again, truck 0 has half the value of the lowest one.

4. *Unit Insertion Profit in Driven Trucks:* Now the probability is proportional to $1+ProfitIns$ if ProfitIns is positive and $\frac{-1}{-1+ProfitIns}$ otherwise, where ProfitIns is the price minus insertion distance in the closest driven truck divided by the capacity demand of the best order. In this way, the truck is favored which has the order which can be most profitable inserted into a driven truck.

Now we have two sets of selection methods to select orders. If we have a non-driven truck, we need to select the most profitable orders to insert into a driven truck, but if the truck was driven, we need to select the worst orders to remove. Hence, if the truck was not driven we have the selection methods random, unit profit, insertion profit and unit insertion profit.

1. *Random:* All orders have the same probability of being selected.

2. *Unit Profit:* The probability is proportional to $1+\frac{Profit}{Capacity}$ if Profit is positive and $\frac{-1}{-1+Profit*Capacity}$ otherwise, where Profit is the prices minus the distance of the sequence and Capacity is the demand of the order sequence. There is no sequence in truck 0. Therefore, the first order is chosen with probability proportional to its price divided by the demand. Then, other orders are added at the beginning or the end of the sequence, where the probability is proportional to the price minus the least distance to the end or beginning divided by the demand of the order. By doing so, the orders that have the most unit profit are chosen to be removed.

3. *Insertion Profit:* The probability is proportional to $1+ProfitIns$ if ProfitIns is positive and $\frac{-1}{-1+ProfitIns}$ otherwise, where ProfitIns is the price of the sequence of orders minus the insertion distance in the non-chosen (driven if possible) truck closest for this sequence. Again, truck 0 needs a different treatment, namely, each order receives a probability proportional to its profit minus the insertion distance and they are drawn until the needed number of orders is reached. In this way, the profitable sequences to insert into other trucks are taken out of the non-driven truck.

4. *Unit Insertion Profit:* Similar to *Insertion Profit* but now divided by the capacity demands of the orders. Hence, the unit wise profitable sequences are removed out of the non-driven truck.

However, if the truck was driven, then we have as selection methods for orders random, distance, average profit and unit average profit.

1. *Random:* All orders have the same probability of being selected.

2. *Distance:* The probability is proportional to the sum of the start and end order of this sequence compared to the order before the start or after the end, respectively. Hence, sequences that are far away from the other orders in the route have a higher probability of being removed.

3. *Average Profit:* The probability is proportional to $\frac{1}{1+Profit}$ if Profit is positive and $1-Profit$ otherwise, where Profit is the price of the sequence of orders minus the distance. By doing so, the orders which have the least profit have the highest probability to be removed.

4. *Unit Average Profit:* Similar to *Average Profit* but now divided by the capacity demands of the orders.

If the orders were from a non-driven truck, then these orders are inserted into the driven truck which is not already chosen and that has the least insertion distance of these orders. However, if the truck was driven, the next truck is the truck, which could be driven or not, which is not yet chosen, does not have an average profit of more than two times the average profit of the removed orders and has the least insertion distance of the orders.
The adaptive mechanism is as follows. If the move yielded a better solution than the current one, then $\zeta = 9$ points are awarded to all selection methods used in this move. After there have been $n^{\mathrm{A}} = 7$ of these Cyclic moves, the probability of each method is adapted by the following formula: $(1-\gamma) * Pr_i + \gamma * Z_i/(n_i^{\mathrm{A}})$, where $\gamma = 0.4$ is the adapting parameter, Pr_i is the current score of selection method i, Z_i is the amount of points collected by selection method i during these n^A moves and n_i^{A} is the number of times that this method is chosen.

3.3.5 Create

If there are some trucks that are not driven, it could be beneficial to fill one truck with profitable orders. The seed of this new truck is determined by giving each non-driven order a score which is the sum of the profit, i.e., price minus distance to this order, of the ten best non-driven orders. The order which has the highest score is chosen as seed. After this, the shifting method is applied on the new truck, where the new truck is the receiver and all the others are the providers.

3.3.6 Destroy

It could also be better to have a truck service another area. The driven truck which will be destroyed is chosen randomly and all the orders in this truck are moved to truck 0. After this, a new route is created using the Create method explained above.

3.3.7 Split

Sometimes, it could be that it is more profitable to create two routes out of one existing. A driven truck is chosen randomly and the route is divided over two trucks where the division is made at the largest distance between orders. The first part remains in the truck and the other part is transferred to a randomly chosen non-driven truck (not truck 0), where the orders which were in the chosen truck are shifted to the virtual truck. After the reallocation, the shifting method is used to refill both affected trucks and the other trucks are providers.

3.3.8 Bomb

Another move is to destruct an area around an order and rebuild it. Randomly an order, which could be driven or not, is picked as the center of the bomb and is moved to the virtual truck. Repeatedly, the closest driven order, compared to the center, is moved to truck 0, until there are at least the lower bound of $Trucks$ affected and at least the ratio $FirstOrders$ of the total driven orders

are moved. In order to avoid affecting too many trucks or orders, the removal is stopped when the upper bound of $Trucks$ are affected or when the ratio $OtherOrders$ of the total driven orders are removed. After this, the shaking method is used, where the receivers are the affected trucks and the providers are the others.

3.3.9 Remove Driven and Insert non-Driven: Jump

These moves are inspired by the ones of Archetti et al. (2007), which removes orders from driven trucks and inserts non-driven orders. A random number of orders, which is drawn from the bounds of $FirstOrders$, is randomly removed from the driven trucks. Then, non-driven orders, i.e., orders which belong to non-driven trucks, are randomly drawn until the price of these orders is at least the price of the orders removed from the driven trucks. Finally, the non-driven orders that were drawn are inserted into driven trucks such that the infeasibility is minimal and the additional insertion distance is minimized. The orders removed from the driven trucks are inserted into the virtual truck. Now the trucks are reevaluated and the infeasibility of the non-driven trucks is removed by randomly moving orders from the infeasible non-driven truck to truck 0 until the infeasible non-driven truck is either feasible or it is driven.

3.3.10 Reseeding

Another destruction move is to remove parts of several routes and rebuild the solution. Randomly a random number of driven trucks, with the bounds of this integer given by $Trucks$, is picked. From each of these trucks, a sequence of a random number of orders, determined by $FirstOrders$ is kept in the truck (this sequence is at most the number of orders minus two) and the other orders are moved to truck 0. After this, the shifting method is used where the adjusted trucks are receivers and the others providers.

3.3.11 TabuSearch

In the final moves, we reset some of the parameters of the Tabu Search, followed by a slightly longer Tabu Search with $N_{\min} \in [200, 300]$, $N_{\max} \in [1000, 1500]$ and $N_{\text{noimpr}} \in [100, 150]$. The other parameters are randomly set to $\lambda_{\text{up}} \in [0.5, 2]$, $\lambda_{\text{down}} \in [0.5, 2]$, $N_{\text{infeas}} \in [3, 5]$, $t_{\text{lower}} \in [4, 9]$ and $t_{\text{upper}} \in [12, 27]$.

3.4 Acceptance decision

We have chosen to accept solutions that improve the current best solutions, i.e., s^{local}, and if this solution is better than the global one, i.e., s^{global}, then the global solution is set to the current solution. To avoid being trapped in a local optimum, we allow a strictly deteriorating solution to be accepted with a certain probability, but only if no improvement of the current best solution was found and there has been no deterioration for a certain number of moves. The probability to accept a deteriorating solution only depends on whether the solution has a cost which is strictly worse and on the amount of moves executed without improvements of the current best solution, i.e., n_{jump}. A solution is never accepted before each move is applied twice on the current solution and a quadratic function is used which increases the probability to jump out of the current solution. This gives the following acceptance criterion for a solution s: s is accepted if $f_1(s) > f_1(s^{\text{local}})$ and a random real number between 0 and 1 is smaller than $\frac{(n_{\text{jump}})^2-(2M_{\max})^2}{2*(4M_{\max})^2}$. When an improvement is found for the local solution or when a deteriorating solution is accepted, n_{jump} is reset to zero or

one, respectively.

3.5 AVNS-S(low)

An overview of the AVNS-Slow is presented in Algorithm 4. The initialization is the same as in the other two heuristics, but is followed by two Tabu Searches with $N_{\min} \in [150, 200]$, $N_{\max} \in [500, 700]$, $N_{\text{noimpr}} \in [30, 60]$, $\lambda_{\text{up}} \in [0.5, 2]$, $\lambda_{\text{down}} \in [0.5, 2]$, $N_{\text{infeas}} \in [3, 6]$, $t_{\text{lower}} \in [5, 10]$ and $t_{\text{upper}} \in [13, 26]$. The other differences are explained in their respective sections.

Algorithm 4 AVNS-Slow

{Initialization phase}
Generate initial solution s^{global} and improve s^{global} by using the Tabu Search heuristic
{Searching phase}
Set $s^{\text{local}} \leftarrow s^{\text{global}}$ and the shaking moves $\mathcal{M} = \{0, ..., M_{\max} - 1\}$
Set $\kappa = 0$, $n_{\text{total}} = 0$, $n_{\text{noimpr}} = 0$ and $n_{\text{jump}} = 0$
while $n_{\text{total}} < N_{\text{total}}$ **and** ($n_{\text{noimpr}} < N_{\text{noimpr}}$ **or** $n_{\text{jump}} < N_{\text{jump}}$) **do**
  if $\kappa == 0$ **then**
    Randomly permutate $\mathcal{M}$
  end if
  {Shaking & Local Search}
  for i=1**to** i=5 **do**
    Set $s^i \leftarrow s^{\text{local}}$, execute move $M(\kappa)$ on s^i and improve s^i by using the Tabu Search heuristic
  end for
  Find the best s^i that is not equal to s^{local} and set s to that solution
  {Acceptance}
  if $f(s) < f(s^{\text{local}})$ **then**
    $s^{\text{local}} \leftarrow s$
    if $f(s) < f(s^{\text{global}})$ **then**
      $s^{\text{global}} \leftarrow s$, $n_{\text{noimpr}} = 0$ and $n_{\text{jump}} = 0$
    else
      if $s == s^{\text{global}}$ **then**
        $n_{\text{jump}} = \max\{M_{\max}, n_{\text{jump}}\}$
      else
        $n_{\text{jump}} \in [M_{\max}/5 + 1, 2M_{\max}/5 + 1]$
      end if
    end if
  else
    if $f(s) > f(s^{\text{local}})$ **and** accept(n_{jump}) **then**
      $s^{\text{local}} \leftarrow s$ and $n_{\text{jump}} \in [M_{\max}/5, 2M_{\max}/5]$
    end if
    $n_{\text{noimpr}} = n_{\text{noimpr}} + 1$ and $n_{\text{jump}} = n_{\text{jump}} + 1$
  end if
  $\kappa = (\kappa + 1) \mod M_{\max}$ and $n_{\text{total}} = n_{\text{total}} + 1$
end while
Return s^{global}

3.5.1 Stopping criterion

Since the AVNS-Slow executes each move 5 times and has less different moves, we use less iterations. The AVNS-Slow stops when either there has been no improvement of the best solution for $N_{\text{noimpr}} = 150$ iterations and the current best solution has not been improved for $N_{\text{jump}} = 2 \times M_{\max} = 34$ iterations, or when the total number of iterations reached $N_{\text{total}} = 750$.

3.5.2 Shaking Moves

Most of the moves are similar to the ones in the AVNS-Fast. However, there are some small changes. First, the solution is copied five times and then the shaking move is applied to each of them, followed by a Tabu Search with the following parameters: $N_{\min}$ is a random integer between $[15, 20]$, $N_{\max}$ between

[166, 233] and N_{noimpr} between [20, 40]. The moves that did change are shortly discussed in their own sections.

Number	Move	#Trucks	#FirstOrders	#OtherOrders	#MaxOrders
1	CyclicDriven	[2,2]	[3,4]	[0,0]	
2	CyclicDriven	[2,2]	[2,4]	[2,4]	
3	CyclicDriven	[2,4]	[3,4]	[3,4]	
4	CyclicDriven	[3,5]	[2,5]	[2,5]	
5	CyclicNotDriven	[2,2]	[3,5]	[2,4]	
6	CyclicNotDriven	[2,3]	[2,4]	[2,4]	
7	CyclicNotDriven	[3,5]	[2,5]	[2,5]	
8	Create				
9	Destroy	50%			
10	Split				
11	Bomb	[2, 4]	[3,5]	[0.1,0.2]	[0.3,0.4]
12	Bomb	[3, 6]	[7,9]	[0.2,0.3]	[0.4,0.5]
13	Jump		[4,9]		
14	Jump		[7,12]		
15	Reseeding	[1,3]	[0.3,0.6]		
16	Reseeding	[4,6]	[0.6,0.7]		
17	TabuSearch				

Table 2: The moves of AVNS-Slow, where the frequency of each move is 1.

3.5.3 Cyclic-Driven and -NotDriven

Instead of using capacity as the fourth selection move for orders in the Cyclic-Driven moves, we use a mixture of the distance in the route and the distance to other routes.

4. *Distance/Average Insertion Distance:* The probability is proportional to the sum of the the distances of the start and the end order of this sequence compared with the order before the first or the order after the last, respectively, divided by the average insertion distance of the sequence in the closest non-chosen driven truck. Hence, sequences that are not close to other orders in this route but are close to other routes are favored.

3.5.4 Destroy

Instead of always following the destruction of a route with the creation of a new one, we have a probability, given by $Trucks$, to apply the create move.

3.5.5 Bomb

Instead of having fixed bounds we have the following random bounds: the minimum amount of trucks is given by a random integer which is drawn from the bounds of $Trucks$, the maximum amount of trucks from $FirstOrders$, the minimum ratio of orders from $OtherOrders$ and the maximum ratio of orders from $MaxOrders$.

3.5.6 Reseeding

Instead of drawing a number of orders, we draw a percentage, given by a random number from the bounds of $FirstOrders$, of the orders as seed.

3.5.7 Tabu Search

Again, the Tabu Search has $N_{\min} \in [150, 200]$, $N_{\max} \in [500, 700]$, $N_{\text{noimpr}} \in [30, 60]$ and the other parameters are reset to $\lambda_{\text{up}} \in [0.5, 2]$, $\lambda_{\text{down}} \in [0.5, 2]$, $N_{\text{infeas}} \in [3, 6]$, $t_{\text{lower}} \in [5, 10]$ and $t_{\text{upper}} \in [13, 26]$.

3.5.8 Acceptance decision

The difference is when to accept a deteriorating solution: The solution s is accepted if $f_1(s) > f_1(s^{\text{local}})$ and a random integer between 0 and 1 is less than $0.5 \times \min\{1, \frac{n_{\text{jump}} - M_{\max}}{M_{\max}}\}$. Furthermore, when a solution is better than the global one, set $n_{\text{jump}} = 0$. However, if the solution is exactly the same as the global one, set $n_{\text{jump}} = \max\{n_{\text{jump}}, M_{\max}\}$. Finally, if the solution was only an improvement to the local solution or it was accepted as a deteriorating move, set $n_{\text{jump}} \in [\frac{M_{\max}}{5} + 1, \frac{2M_{\max}}{5} + 1]$.

4 Computational study

In this section, we present the results of the R-TS, where a part comes form Huijink et al. (2014), the AVNS-Fast and the AVNS-Slow and compare it with the heuristics from the literature. Our heuristics are coded using Microsoft Visual Studio Express 2013 for Windows Desktop in the language c++ on a laptop equipped with Intel i5-2540 2.60GHZ and 4GB RAM running on Windows 7 SP1. Note that this laptop has two cores and uses multithreading, i.e., each physical core is split into two logical ones. However, the code was not parallelized and since other programs were running on the same laptop, it is unlikely that the program did fully utilize a core. This, together with the differences in computers used, languages, and maybe parallelization make it hazardous to compare the running times.

4.1 Benchmark instances

The test instances of Bolduc et al. (2008) are modifications of the test instances of Christofides et al. (1979) (CE) and Golden et al. (1998) (G). In the CE instances, the number of orders ranges from 50 to 199, the number of trucks from 4 to 13, and the number of orders divided by trucks ($\bar{n}$) from 8.3 to 20.0. For the G instances, the number of orders ranges from 200 to 480, the number of trucks from 4 to 29, and the number of orders divided by trucks from 12.0 to 60.0. Note that in the G instances the first 12 instances have a large $\bar{n}$, while the last 8 have a $\bar{n}$ between 12.0 and 16.6. The coordinates, demand and capacity are kept, but all the time restrictions, i.e., route-length, time-windows and fixed time per stop, are dropped. Additionally, the number of available trucks is set to $\lceil 0.8 \sum_{i=1}^{n} q_i \rceil$, i.e., there are just enough trucks such that 80% of the orders can be delivered by the private fleet, and each truck has a variable cost of 1. Furthermore, the fixed cost F is the average route length within the best known solution of the original instance with timing restrictions, rounded to the nearest integer which is divisible by 20. Finally, the outsource price of each order is set as follows: Let $\bar{n}$ be the total number of orders divided by the number of trucks and let $q_{\min}, q_{\max}$ the minimum and maximum demand of the orders, respectively. Set $\eta = \frac{q_{\max} - q_{\min}}{3}$ and

$$\mu_i = \begin{cases} 1 & \text{if } q_i \in [q_{\min}, q_{\min} + \eta), \\ 1.5 & \text{if } q_i \in [q_{\min} + \eta, q_{\min} + 2\eta), \\ 2 & \text{if } q_i \in [q_{\min} + 2\eta, q_{\max}]. \end{cases}$$

The outsource costs, or common carrier costs, are given by $p_i = round(\frac{F}{\bar{n}} + \mu_i \cdot d_{0i})$, i.e., the common carrier costs consist of an estimation of the fixed costs and a factor times the distance from the depot to the order. An overview of

these test-instances is presented in Bolduc et al. (2008). Due to this pricing, the test-instances have the characteristic that the outsourcing is done out of necessity instead of efficiency. This is reflected by the fact that the AVNS-Slow, which has a average gap of 0.06% on the homogenous instances, uses all the available trucks in 29 of these 34 instances. Moreover, the AVNS-Slow delivers on average 83% of the total capacity of the orders. To amend the issue that the outsourcing is only done out of necessity, we modified the test instances by halving the outsource prices. This set of instances, called the Half-Original instances, are copies of the original instances of Bolduc et al. (2007), but now the outsource costs are halved, i.e., $p_i^{half} = \frac{p_i}{2}$. Furthermore, the instances with the original outsource pricing, and by inheritance the Half-Original pricing, have as additional characteristic that small orders are preferred upon larger ones. The AVNS-Slow delivers on average 90% of the orders which only have 83% of the total capacity of the orders and this difference is even larger for the Half-Orig pricing. Why this is the case is illustrated in the following example.

Example 4.1. *Suppose that we have four orders, where the characteristics are given in Figure 1 with (number, demand, outsource price), and two trucks each with capacity* 44*. The total distance when delivering all orders is* 25*, as is*

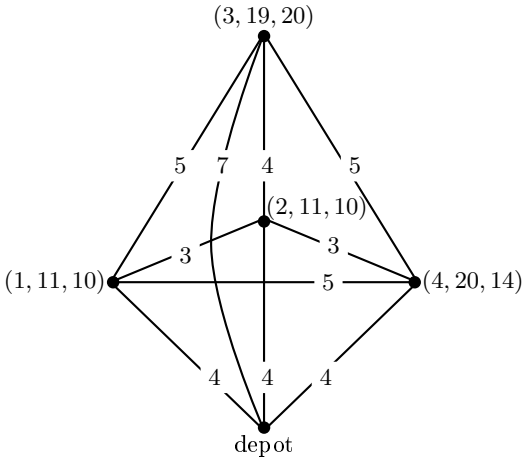

Figure 1: The orders and distances.

illustrated in Table 3. This gives, according to the settings in the original pricing, that the (rounded) fixed costs per truck are 12 *and the outsource prices* $p_i = 6+\mu_i \cdot d_{0i}$*. This gives the following three best solutions together with the solutions when everything is either outsourced or driven.*

Orders Driven	Trucks	Distance	OutsourceCosts	Total Costs
∅	0	0	54	54
{3,4}	1	16	20	48
{1,2,3}	1	17	14	43
{1,2,4}	1	20	10	46
{1,2,3,4}	2	25	0	49

Table 3: The basic solutions and the three best.

This implies that delivering orders $\{1, 2, 3\}$ *is the best one can do, i.e., choosing two smaller orders (1 and 2) instead of one larger (4) even while the distance*

to deliver both 1 and 2 is larger. This is due to the fixed cost $\frac{F}{\bar{n}} = 6$ *in the formula of the pricing together with that the other part of the pricing, i.e.,* $\mu_i d_{0i}$*, cannot be more than twice as large for orders that have the same distance to the depot. In this case, order 4 has also the lowest ratio of outsourcing price divided by demand. However, the following modification presented in Figure 2, where order 4 is further away and has the highest price, shows that there are also simple examples where smaller orders are more profitable in cases where the outsourced order has the highest ratio. The total distance when delivering*

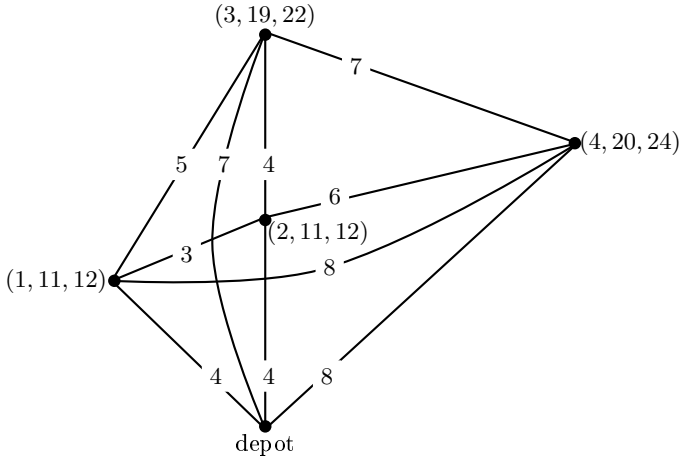

Figure 2: The orders and distances.

all orders is now 33*, see Table 4. Hence, the (rounded) fixed costs per truck are* 16 *and the outsource prices are also slightly adjusted. The numbers in the three solutions also change and are given below.*

Orders Driven	Trucks	Distance	OutsourceCosts	Total Costs
∅	0	0	70	70
{3,4}	1	22	24	62
{1,2,3}	1	17	24	57
{1,2,4}	1	21	22	59
{1,2,3,4}	2	33	0	65

Table 4: The basic solutions and the three best of the modified example.

△

To amend the issue that large orders are more likely to be outsourced and to avoid that the outsource price is dependent on the distance to the depot, we created a third set of prices. This set of instances, called the New instances, again copies the original instances, but now with the outsource costs as a proportion of the fixed costs that the order capacity-wise consumes times a parameter which represents the additional costs (rounded to quarters), i.e., $p_i^{new} = \frac{1}{4} round(4 \cdot 1.1 \cdot F \frac{q_i}{Q})$. These new costs are still slightly dependent on the distance since the fixed costs F is present in this formula and since F is derived from the best known solution in the original problem with timing restrictions. Note that we first discussed these half and new pricing mechanisms in Huijink et al. (2014).

4.2 Results

As is common practice in the literature, we ran our heuristics 10 times and recorded the best and average solution. A summary of the results is displayed in Tables 5-8, while the detailed results can be found in Tables 9-13 in A. For all methods, unless stated otherwise below, "best" corresponds to the average deviation to the best known solutions (BKS) of the best found solution over the 10 runs, "avg" corresponds to the average deviation to the best known solutions of the average solution of the 10 runs, "CPU(s)" corresponds to the average running time of the 10 runs, and "#BKS" corresponds to the number of times the method found the best known solution. A few notes on the heuristics in the literature. For some heuristics only the best found solution is reported while only one run is provided of the RIP (Bolduc et al., 2008). Furthermore, the heuristics of Côté and Potvin (2009) and Potvin and Naud (2011) use truncated coordinates (Côté et al., 2014) which implies that only a part of their results can be compared. Moreover, the Tabu Search (TS) of Potvin and Naud (2011) uses the same code as in Côté and Potvin (2009) (TS25), but now with 50000 iterations instead of the 25000 as in Côté and Potvin (2009). Although the number of iterations have been doubled, the results of the TS in Potvin and Naud (2011) are much worse. Finally, Kratica et al. (2012) reported the best result found over 20 runs instead of the commonly used 10 and the values of the AVNS-RN are reverse engineered since Stenger et al. (2013a) only provides the current best-known solution and a gap. Therefore, the exact values of the AVNS-RN in Table 9 could be slightly different, but the influence on the gaps is negligible.

		RIP	TS25	TS	TS+	GA	AVNS	AVNS-RN	MS-LS	MS-ILS	UHGS	R-TS	AVNS-F	AVNS-S
CE	best	1.09%	0.18%	0.45%	0.34%	*4.42%*	0.22%	0.18%	1.43%	0.09%	0.04%	0.25%	0.07%	0.04%
	avg	-	0.43%	-	-	-	-	0.43%	2.73%	0.33%	0.17%	0.80%	0.32%	0.21%
	CPU(s)	178	50	119	125	*154*	177	174	19	368	615	80	167	479
	#BKS	0	5	4	4	*0*	3	7	0	8	11	4	8	11
G	best	2.21%	*0.43%*	*0.61%*	*0.58%*	*6.43%*	0.0.86%	0.57%	2.87%	0.69%	0.33%	0.89%	0.25%	0.07%
	avg	-	*0.68%*	-	-	-	-	1.40%	3.76%	1.22%	0.64%	1.66%	0.61%	0.44%
	CPU(s)	1655	*485*	*1070*	*2820*	*7774*	1094	1116	180	1437	2262	910	2349	7580
	#BKS	0	*0*	*0*	*0*	*0*	0	0	0	0	7	0	1	6
Avg	best	1.75%	*0.27%*	*0.51%*	*0.43%*	*5.60%*	0.59%	0.57%	2.28%	0.44%	0.21%	0.63%	0.18%	0.06%
	avg	-	*0.52%*	-	-	-	-	1.01%	3.33%	0.86%	0.44%	1.30%	0.49%	0.34%
	CPU(s)	1047	*208*	*465*	*1105*	*4636*	717	728	113	997	1584	568	1451	4656

Table 5: Comparison on homogenous instances and original pricing.

For the homogenous instances with original pricing, the AVNS-Slow has a gap of 0.06% while the UHGS has 0.21% and the AVNS-Fast has 0.18%. Hence, the AVNS-Slow closes the gap by over 71%. For the average solutions the gaps are closer to each other, 0.34% for the AVNS-Slow against 0.44% for the UHGS and 0.49% for the AVNS-Fast. This implies that the AVNS-Fast is comparable with the UHGS, which was the heuristic for the VRPPC and CPTP before ours with the lowest gap (Vidal et al., 2014), but the AVNS-Fast finds fewer best known solutions. Furthermore, note that the best solution of the UHGS sometimes has a rather large gap of 1.73% (at G-12), while the largest gap of AVNS-Fast is 0.81% (at G-07), and finally, for the AVNS-Slow it is only 0.29% (at G-10). Moreover, the AVNS-Slow outperforms the UHGS on the G instances with many orders per truck (G-01 up to G-12), while the UHGS is slightly better at the instances with many trucks. Finally, 2 new BKS solutions where found for the CE instances and 12 for the G instances. The results show that using different neighborhoods can be worthwhile since our AVNS-Fast is much better than the AVNS(-RN) of Stenger et al. (2013a,b).

		RIP	TS	TS+	GA	R-TS	AVNS-F	AVNS-S
CE	best	0.83%	0.58%	0.47%	*3.83%*	0.74%	0.31%	0.05%
	avg	-	-	-	-	1.57%	0.99%	0.47%
	CPU(s)	181	118	126	*156*	86	149	556
	#BKS	0	1	1	*0*	1	2	9
G	best	2.08%	*1.50%*	*0.53%*	*6.73%*	1.49%	0.45%	0.02%
	avg	-	-	-	-	2.53%	1.03%	0.54%
	CPU(s)	1658	*1358*	*4514*	*7745*	829	2635	8573
	#BKS	0	*0*	*1*	*0*	0	1	19
Avg	best	1.56%	*0.94%*	*0.49%*	*5.53%*	1.18%	0.39%	0.03%
	avg	-	-	-	-	2.03%	1.02%	0.51%
	CPU(s)	1050	*603*	*1843*	*4620*	523	1611	5272

Table 6: Comparison on heterogenous instances and original pricing.

The AVNS-Slow is better than the AVNS-Fast in the heterogenous case with a bigger difference than in the homogenous case, 0.04% against 0.39%. We found 8 new best known solutions for the CE instances and 19 for the G instance. Furthermore, the AVNS-Slow is much better than the other heuristics. Table 7 shows the summary of the results of our three heuristics on the instances with half-original pricing and Table 8 on the instances with the new pricing. The results on these instances are in line with what is found on the original instances, namely, the G instances seem to be harder than the CE instances and the ANVS-Slow is the best heuristics, although it comes at a price in running time, and is closely followed by the AVNS-Fast. Finally, in Table 14 we present the

		homogenous			heterogenous		
		R-TS	AVNS-F	AVNS-S	R-TS	AVNS-F	AVNS-S
CE	best	0.19%	0.06%	0.00%	0.68%	0.10%	0.03%
	avg	0.63%	0.18%	0.10%	1.86%	0.36%	0.25%
	CPU(s)	54	111	443	57	106	480
	#BKS	8	10	12	2	8	11
G	best	1.05%	0.11%	0.04%	1.51%	0.19%	0.04%
	avg	1.56%	0.50%	0.33%	2.38%	0.70%	0.44%
	CPU(s)	866	2267	6901	761	2683	7434
	#BKS	0	4	13	0	4	15
Avg	best	0.70%	0.09%	0.03%	1.19%	0.15%	0.03%
	avg	1.19%	0.37%	0.24%	2.20%	0.57%	0.37%
	CPU(s)	533	1382	4253	472	1624	4583

Table 7: Comparison on half-original pricing.

		homogenous			heterogenous		
		R-TS	AVNS-F	AVNS-S	R-TS	AVNS-F	AVNS-S
CE	best	0.24%	0.07%	0.02%	0.38%	0.12%	0.02%
	avg	0.74%	0.25%	0.10%	0.91%	0.36%	0.23%
	CPU(s)	55	113	394	55	100	442
	#BKS	3	8	10	1	4	11
G	best	0.48%	0.16%	0.01%	0.66%	0.12%	0.01%
	avg	0.86%	0.50%	0.21%	1.08%	0.42%	0.23%
	CPU(s)	731	2085	5857	649	1989	5425
	#BKS	0	0	18	0	8	16
Avg	best	0.39%	0.13%	0.01%	0.55%	0.12%	0.01%
	avg	0.83%	0.37%	0.17%	1.02%	0.40%	0.23%
	CPU(s)	454	1276	3617	406	1214	3385

Table 8: Comparison on new pricing.

number of times, averaged over the ten runs of AVNS-Slow, that a move found

an improvement on the G instances with new pricing. Furthermore, the average improvement of each move can be found in Table 15. These two tables show that the destroy move is, compared to the other moves, less promising. Additionally, while both the create and the split move do not often find an improvement, they are important for some instances (G-12 and G-16) and often result in high gains compared to the other moves. Note that the results could be slightly skewed since most moves will find an improvement at the start of the AVNS-Slow.

5 Conclusions

We introduced several new neighborhoods and two heuristics to exploit these different neighborhoods. The heuristics are tested on instances in the literature and two new sets of instances. Computational experiments show that the new neighborhoods yield solutions that are significantly better than the ones obtained by other heuristics. Although the improvement in absolute numbers is small, 0.21% for the UHGS and 0.06% for our AVNS-Slow for the homogenous instances with original pricing, the relative improvement is a quite impressive 71%. Furthermore, many best known solutions have been improved. Note that the results are more pronounced in the heterogenous case and most of the best known solutions are improved. While our results are impressive, they were obtained without optimizing the different parameters. Therefore, additional research is needed to optimize the parameters. One could also research which neighborhoods are the most rewarding and new neighborhoods could be created, especially for instances with many trucks and a few orders per truck. Although our heuristic is quite good when the fleet is heterogenous, we did not include a move for this case, e.g., switching the trucks on routes. Finally, it is remarkable that a variable neighborhood search is more effective when several different neighborhoods are used.

This novel way of using many different neighborhoods could be improved by creating moves to handle many trucks with a few orders, creating a score method for different moves similarly as for the selection methods in the neighborhoods of the Cyclic moves, to name a few. Furthermore, several speeding up techniques could be applied to the heuristics, e.g., using integers instead of doubles, parallelization of the code or using estimations or bounds to find promising moves. Other areas of research are adding time-windows or other practical restrictions to the heuristics.

Acknowledgements

This research was partly supported by the Dutch Institute for Advanced Logistics under the project 4C4D, which is gratefully acknowledged.

References

C. Archetti, A. Hertz, and M. G. Speranza. Metaheuristics for the team orienteering problem. *Journal of Heuristics*, 13(1):49–76, 2007.

C. Archetti, D. Feillet, A. Hertz, and M. G. Speranza. The capacitated team orienteering and profitable tour problems. *Journal of the Operational Research Society*, 60:831–842, 2009.

M.-C. Bolduc, J. Renoud, and F. Boctor. A heuristic for the routing and carrier

selection problem. *Europian Journal of Operational Research*, 183:926–932, 2007.

M.-C. Bolduc, J. Renoud, F. Boctor, and G. Laporte. A pertubation metaheuristic for the vehicle routing problem with private fleet and common carriers. *Journal of the Operational Research Society*, 59:776–787, 2008.

S. Boussier, D. Feillet, and M. Gendrau. An exact algorithm for team orienteering problems. *4OR: A quarterly Journal of Operations Research*, 5:211–230, 2007.

N. Christofides, A. Mingozzi, and P. Toth. The Vehicle Routing Problem. In N. Christofides, A. Mingozzi, P. Toth, and C. Sandi, editors, *Combinatorial Optimization*, pages 315–338. Wiley, Chichester, 1979.

C.-W. Chu. A heuristic algorithm for the truckload and less-than-truckload problem. *European Journal of Operational Research*, 165:657–667, 2005.

J.-F. Côté and J.-Y. Potvin. A tabu search heuristic for the vehicle routing problem with private fleet and common carrier. *European Journal of Operational Research*, 198:464–469, 2009.

J.-F. Côté, T. Vidal, and F. Santos. Communication by email. 2014.

D. Feillet, P. Dejax, and M. Gendreau. Traveling Salesman Problems with Profits. *Transportation Science*, 39(2):188–205, 2005.

B. Golden, E. Wasil, J. Kelly, and I.-M. Chao. The impact of metaheuristics on solving the vehicle routing problem: Algorithms, problem sets, and computational results. In T. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 33–56. Springer US, 1998. ISBN 978-0-7923-8161-7.

R. W. Hall and M. Racer. Transportation with common carrier and private fleets: system assignment and shipment frequency optimization. *IIE Transactions*, 27(2):217–225, 1995.

S. Huijink, G. Kant, and M. Peeters. Analysis of construciton methods for the vehicle routing problem with private fleet and common carrier. In *Proceedings of the ILS 2014 conference, Breda*, pages 1–8, 2014.

J. Kratica, T. Kostić, D. Tošić, D. Dugošija, and V. Filipović. A genetic algorithm for the routing and carrier selection problem. *Computer Science and Information Systems*, 9(1):49–62, 2012.

J.-Y. Potvin and M.-A. Naud. Tabu search with ejection chains for the vehicle routing problem with private fleet and common carrier. *Journal of the Operational Research Society*, 62:326–336, 2011.

J. Renaud, F. F. Boctor, and G. Laporte. A fast composite heuristic for the symmetric traveling salesman problem. *INFORMS Journal on Computing*, 8 (2):134–143, 1996.

A. Stenger, M. Schneider, and D. Goeke. The prize-collecting vehicle routing problem with single and multiple depots and non-linear cost. *EURO Journal on Transportation and Logistics*, pages DOI: 10.1007/s13676–013–0022–4, 2013a.

A. Stenger, D. Vigo, S. Enz, and M. Schwind. An adaptive Variable Neighborhood Search Algorithm for a Vehicle Routing Problem Arising in Small Package Shipping. *Transportation Science*, 47:64–80, 2013b.

L. Tang and X. Wang. Iterated local search algorithm based on very large−scale neighborhood for prize-collecting vehicle routing problem. *The International Journal of Advanced Manufacturing Technology*, 29:1246–1258, 2006.

L. Tang, X. Zhang, and Q. Guo. Two Hybrid Metaheuristic Algorithms for Hot Rolling Scheduling. *ISIJ International*, 46(4):529–538, 2009.

T. Vidal, N. Maculan, L. S. Ochi, and P. H. V. Penna. Large neighborhoods with implicit customer selection for vehicle routing problems with profits. *Working Paper, MIT*, 2014:arXiV:1401.3794v2, 2014.

A The detailed results

		RIP		TS25			TS+		TS		GA		AVNS		AVNS-RN		
		Bolduc et al. (2008)		Côté and Potvin (2009)			Potvin and Naud (2011)		Potvin and Naud (2011)		Kratica et al. (2012)		Stenger et al. (2013b)		Stenger et al. (2013a)		
Instance	BKS	single	CPU(s)	best	avg	CPU(s)	best	CPU(s)	best	CPU(s)	best	CPU(s)	best	CPU(s)	best	avg	CPU(s)
CE-01	1119.47	1132.91	25.0	1119.47	1119.47	8.5	1119.47	24.9	1119.47	24.3	1158.98	49.7	1123.95	92.5	1119.47	1124.06	81.2
CE-02	1814.52	1835.76	73.0	1814.52	1816.07	12.5	1814.52	33.9	1814.52	33.0	1893.66	91.2	1814.52	48.6	1814.52	1816.88	63.4
CE-03	1919.05	1959.65	107.0	1924.99	1930.28	34.7	1930.66	81.0	1921.10	78.6	1987.75	111.4	1920.86	212.1	1919.05	1931.52	258.1
CE-04	2505.39	2545.72	250.0	2515.50	2526.41	83.3	2525.17	200.6	2525.17	193.2	2668.87	204.1	2512.05	279.7	2509.20	2526.00	179.6
CE-05	3081.59	3172.22	474.0	3097.99	3112.25	128.3	3117.10	353.2	3113.58	309.9	3279.64	342.1	3099.77	228.6	3111.61	3121.21	132.9
CE-06	1207.47	1208.33	25.0	1207.47	1207.47	9.9	1207.47	25.2	1207.47	25.5	1233.20	47.3	1207.81	75.9	1207.47	1208.80	79.8
CE-07	2004.53	2006.52	71.0	2006.52	2010.96	14.0	2006.52	34.0	2006.52	32.7	2086.17	92.0	2013.93	50.9	2004.53	2008.74	61.0
CE-08	2052.05	2082.75	110.0	2055.64	2063.06	36.9	2056.59	81.6	2060.17	85.1	2130.82	113.8	2052.05	253.1	2052.05	2062.31	251.1
CE-09	**2419.84**	2443.94	260.0	2429.19	2433.86	83.3	2435.97	188.2	2438.43	185.3	2558.70	224.7	2432.51	259.0	2431.22	2439.94	190.8
CE-10	**3373.84**	3464.90	478.0	3393.41	3402.72	129.6	3401.83	345.7	3406.82	311.1	3598.36	358.7	3391.35	201.0	3389.09	3407.02	162.2
CE-11	2330.94	2333.03	195.0	2330.94	2336.59	54.6	2332.36	131.0	2353.39	126.3	2383.34	137.4	2332.21	316.0	2330.94	2332.34	370.5
CE-12	1952.86	1953.55	128.0	1952.86	1961.49	24.2	1952.86	59.5	1952.86	60.4	2042.84	111.0	1953.55	92.9	1953.64	1953.64	107.5
CE-13	2858.83	2864.21	188.0	2859.12	2863.96	53.7	2860.89	132.1	2882.70	130.0	2929.02	163.7	2858.94	278.5	2858.94	2860.94	351.4
CE-14	2213.02	2224.63	110.0	2214.14	2220.23	24.8	2219.97	64.2	2219.97	65.0	2338.22	103.8	2215.38	93.2	2215.45	2215.45	148.8
Avg		1.09 %	178.1	0.18 %	0.43 %	49.9	0.34 %	125.4	0.45 %	118.6	4.42 %	153.6	0.22 %	177.3	0.16 %	0.44 %	174.2
G-01	**14111.95**	14388.58	651.0								14910.52	629.3	14157.08	652.6	14155.86	14178.46	979.3
G-02	**19140.69**	19505.00	1178.0								20258.91	4568.4	19204.36	1558.4	19187.73	19283.51	1809.6
G-03	**24368.29**	24978.17	2061.0								25941.17	13529.3	24602.61	2356.1	24535.87	24706.75	2267.4
G-04	**34231.56**	34957.98	3027.0								36083.77	22360.7	34415.82	2500.9	34535.60	34682.98	2241.1
G-05	14223.63	14683.03	589.0								14875.44	1803.1	14272.32	1301.1	14276.20	14318.88	2263.5
G-06	**21357.16**	22260.19	1021.0								22440.03	4826.8	21440.79	1783.5	21396.38	21529.04	2049.0
G-07	**23263.22**	23963.36	1628.0								24621.42	11098.2	23375.60	2262.8	23434.04	23607.02	2107.7
G-08	**29657.38**	30496.18	2419.0								31326.38	12532.0	29797.62	2339.7	29819.94	29920.96	1907.3
G-09	**1319.72**	1341.17	832.0	1323.57	1324.87	274.7	1325.62	819.1	1328.14	611.0	1368.47	3236.9	1335.45	602.0	1332.04	1341.42	704.1
G-10	**1583.50**	1612.09	1294.0	1592.93	1598.14	471.2	1590.82	1762.3	1590.83	938.8	1646.20	7682.2	1604.50	978.4	1604.41	1615.02	864.9
G-11	**2159.78**	2198.45	2004.0	2166.66	2174.45	720.2	2173.80	3284.3	2172.28	1492.7	2235.24	17381.0	2189.02	1534.3	2188.65	2205.53	1311.0
G-12	2479.62	2521.79	2900.0	2490.01	2499.80	1071.4	2495.02	8587.6	2492.75	2309.7	2578.12	32100.7	2520.29	2043.9	2528.72	2542.11	1975.8
G-13	2258.02	2286.91	802.0	2271.29	2274.13	157.6	2274.12	504.5	2278.99	360.8	2347.49	1113.6	2291.83	116.5	2283.74	2294.41	170.8
G-14	2683.73	2750.75	1251.0	2693.35	2702.50	248.4	2703.31	976.9	2705.00	610.4	2796.74	2454.8	2708.22	183.5	2717.77	2728.54	215.6
G-15	3145.11	3216.99	1862.0	3157.31	3162.85	377.1	3161.26	1952.0	3158.92	924.8	3283.07	5083.7	3194.82	357.3	3177.52	3198.96	221.0
G-16	3620.71	3693.62	2778.0	3637.52	3645.80	556.4	3638.39	4675.1	3639.11	1313.7	3804.04	11131.3	3671.34	561.2	3674.68	3689.94	370.0
G-17	1666.31	1701.58	806.0								1898.36	372.9	1682.49	110.3	1672.02	1694.55	182.3
G-18	2730.55	2765.92	1303.0								3079.03	851.8	2741.80	156.4	2751.63	2761.05	211.5
G-19	**3494.27**	3576.92	1903.0								3940.71	1110.9	3507.94	194.1	3516.85	3530.97	234.3
G-20	**4306.85**	4378.13	2800.0								4823.76	1606.5	4332.44	290.3	4347.85	4358.94	241.3
Avg		2.21 %	1655.5	0.43 %	0.68 %	484.6	0.58 %	2820.2	0.61	1070.2	6.43 %	7773.7	0.86 %	1094.2	0.86 %	1.40	1116.4
TAvg		1.75 %	1047.1	0.27 %	0.52 %	208.0	0.43 %	1105.3	0.51	464.7	5.60 %	4636.0	0.59 %	716.6	0.57 %	1.01	728.4

Table 9: Comparison of heuristics in the literature on homogenous instances with original pricing: Part 1.

[1]Note that the results from the Tabu Searches of Côté and Potvin (2009) (TS25) and Potvin and Naud (2011) (TS and TS+) are omitted for G-(H-)-01 up to G-(H-)-07 and from G-(H-)-17 to G-(H-)-20. This is because the heuristics of Côté and Potvin (2009) and Potvin and Naud (2011) accidently truncate the coordinates to integers when loading the instances (Côté et al., 2014). The truncation implies that those instances are different and hence they cannot be compared.

		MS-LS Vidal et al. (2014)			MS-ILS Vidal et al. (2014)			UHGS Vidal et al. (2014)			R-TS Huijink et al. (2014)			ANVS-Fast this paper			AVNS-Slow this paper		
Instance	BKS	best	avg	CPU(s)	best	avg	CPU(s)	best	avg	CPU(s)	best	avg	CPU(s)	best	avg	CPU(s)	best	avg	CPU(s)
CE-01	1119.47	1121.32	1128.28	5.2	1119.47	1119.66	34.1	1119.47	1119.66	38.5	1119.47	1120.15	10.1	1119.47	1119.47	16.2	1119.47	1119.47	47.4
CE-02	1814.52	1840.71	1883.44	3.9	1814.52	1817.34	68.8	1814.52	1815.63	59.6	1814.52	1821.07	24.0	1814.52	1817.89	40.5	1814.52	1817.06	95.8
CE-03	1919.05	1943.64	1958.80	17.0	1922.18	1929.60	258.1	1919.05	1922.88	476.8	1921.20	1930.42	50.9	1921.10	1928.96	94.0	1919.05	1925.08	236.7
CE-04	2505.39	2548.29	2568.49	26.3	2505.39	2516.12	576.7	2505.39	2509.82	934.7	2520.88	2529.49	122.4	2506.40	2521.85	265.1	2509.81	2518.14	642.7
CE-05	3081.59	3181.86	3201.29	29.3	3090.53	3102.95	620.7	3081.59	3095.58	1289.3	3096.61	3110.45	234.5	3086.88	3102.68	496.4	3090.49	3101.40	1437.2
CE-06	1207.47	1207.47	1216.57	5.8	1207.47	1207.56	36.5	1207.47	1207.47	38.5	1207.47	1208.43	11.2	1207.47	1207.56	15.0	1207.47	1207.47	44.2
CE-07	2004.53	2046.62	2079.67	4.1	2006.52	2022.93	71.2	2006.52	2012.33	73.2	2004.88	2011.64	26.1	2004.53	2009.03	33.0	2004.53	2009.58	98.6
CE-08	2052.05	2088.10	2100.59	17.3	2054.64	2062.21	263.4	2052.05	2057.57	500.3	2065.16	2072.62	51.0	2058.22	2068.52	111.4	2052.05	2059.56	239.6
CE-09	**2419.84**	2478.01	2505.24	25.9	2428.03	2433.28	482.1	2424.32	2428.19	1241.0	2438.35	2447.73	134.0	2425.41	2431.26	309.2	2420.71	2426.35	953.9
CE-10	**3373.84**	3462.56	3491.59	30.1	3382.23	3393.78	714.9	3381.67	3387.12	1229.9	3399.95	3413.53	219.5	3380.43	3392.14	554.5	3373.84	3388.22	1704.1
CE-11	2330.94	2343.03	2408.13	37.5	2330.94	2336.06	891.7	2330.94	2331.13	1202.9	2330.94	2392.91	68.7	2330.94	2338.09	132.1	2330.94	2330.94	399.0
CE-12	1952.86	1970.05	1982.06	8.7	1952.86	1953.13	113.6	1952.86	1953.13	150.8	1953.55	1966.32	47.5	1952.86	1953.13	85.1	1952.86	1952.86	183.6
CE-13	2858.83	2909.83	3025.26	40.1	2858.83	2859.01	881.6	2858.83	2859.07	1184.9	2858.94	2892.05	75.8	2858.83	2859.19	130.9	2858.83	2858.92	437.8
CE-14	2213.02	2215.38	2226.44	9.7	2213.02	2213.02	144.8	2213.02	2213.02	189.8	2214.11	2214.32	42.2	2213.02	2214.24	55.2	2213.02	2213.78	185.2
Avg		1.43 %	2.73 %	18.6	0.09 %	0.33 %	368.4	0.04 %	0.17 %	615.0	0.25 %	0.80 %	79.8	0.07 %	0.32 %	167.0	0.04 %	0.21 %	479.0
G-01	**14111.95**	14272.27	14329.96	102.5	14151.74	14165.45	1811.6	14131.18	14151.51	2405.9	14215.96	14255.60	327.2	14159.16	14178.79	754.8	14129.48	14163.43	2433.9
G-02	**19140.69**	19417.12	19524.50	209.6	19142.75	19191.56	1828.1	19166.58	19190.77	2409.9	19427.81	19570.07	701.9	19145.09	19212.59	1573.0	19140.69	19254.23	6630.0
G-03	**24368.29**	24916.33	25038.41	415.2	24493.16	24609.36	1828.8	24409.02	24588.29	2418.1	24815.06	25175.22	1282.0	24435.85	24584.51	3224.4	24406.67	24566.00	11062.8
G-04	**34231.56**	34883.27	35182.78	639.2	34708.93	34907.49	1864.6	34362.80	34517.47	2421.1	34650.73	35012.77	1853.4	34285.64	34485.85	4939.1	34231.56	34425.00	15875.4
G-05	14223.63	14492.24	14735.12	199.1	14255.09	14373.87	1817.4	14223.63	14296.07	2408.0	14398.20	14627.22	311.4	14223.63	14305.10	414.3	14229.50	14261.06	1591.3
G-06	**21357.16**	21741.15	22024.07	285.3	21382.16	21546.18	1834.2	21396.60	21488.29	2411.4	21624.94	21979.24	656.2	21509.52	21564.90	825.5	21357.16	21440.38	4337.2
G-07	**23263.22**	23751.10	23980.00	381.3	23407.50	23547.12	1830.5	23373.38	23463.05	2414.9	23807.31	24096.44	1039.2	23452.22	23570.53	2156.5	23263.22	23440.51	7585.3
G-08	**29657.38**	30271.82	30459.11	447.7	29953.21	30064.28	1856.4	29823.18	29918.06	2415.6	29960.57	30181.65	1694.4	29717.00	29874.71	3917.1	29657.38	29864.19	12316.5
G-09	**1319.72**	1370.26	1397.08	60.7	1332.09	1339.06	1305.4	1328.65	1332.63	2323.4	1323.25	1328.30	458.9	1321.20	1324.08	1211.1	1320.29	1325.79	3852.0
G-10	**1583.50**	1664.96	1682.31	95.9	1595.45	1617.58	1709.2	1597.61	1603.82	2342.3	1586.47	1595.64	684.9	1583.78	1590.01	2096.5	1588.05	1592.14	6922.9
G-11	**2159.78**	2248.04	2281.79	163.3	2196.75	2228.23	1811.6	2182.01	2192.68	2405.2	2167.57	2180.98	1203.8	2164.63	2173.27	3850.1	2163.50	2172.45	12303.3
G-12	2479.62	2624.19	2652.57	236.9	2540.92	2553.40	1818.0	2522.64	2529.84	2407.2	2493.57	2511.72	2074.2	2480.21	2493.22	7812.0	2483.06	2493.94	19555.0
G-13	2258.02	2319.74	2337.43	29.2	2274.19	2277.57	513.1	2258.02	2261.50	1412.7	2273.85	2284.81	218.3	2266.43	2272.29	523.2	2261.66	2264.92	2260.2
G-14	2683.73	2764.11	2791.23	43.5	2701.78	2708.56	917.7	2683.73	2687.50	1935.7	2694.86	2710.24	396.6	2694.34	2699.00	1163.7	2684.66	2689.99	4838.5
G-15	3145.11	3272.34	3296.86	61.8	3170.50	3177.53	1480.3	3145.11	3152.00	2301.4	3168.79	3174.45	1014.1	3151.41	3161.40	2344.9	3150.67	3156.84	8198.0
G-16	3620.71	3794.17	3811.80	89.7	3641.69	3672.62	1755.0	3620.71	3632.04	2450.1	3641.54	3683.87	1086.1	3633.69	3645.61	3558.9	3624.56	3633.76	12741.7
G-17	1666.31	1708.26	1717.55	19.4	1669.59	1677.37	379.0	1666.31	1671.72	1805.3	1676.52	1683.68	365.5	1666.96	1674.11	782.5	1666.31	1672.81	1878.0
G-18	2730.55	2793.63	2801.70	27.7	2734.81	2741.10	582.7	2730.55	2733.12	2035.0	2737.70	2749.69	604.5	2735.65	2739.57	1144.7	2731.28	2734.86	3916.6
G-19	**3494.27**	3570.94	3585.64	36.6	3508.53	3515.47	712.9	3497.20	3504.26	1989.5	3530.26	3538.79	961.2	3502.05	3509.68	1943.4	3494.28	3499.55	5225.4
G-20	**4306.85**	4405.90	4433.41	45.9	4316.28	4333.59	1079.2	4312.45	4319.37	2523.0	4347.73	4375.36	1267.8	4320.54	4331.77	2750.6	4307.63	4314.48	8081.6
Avg		2.87 %	3.76 %	179.5	0.69 %	1.22 %	1436.8	0.33 %	0.64 %	2261.8	0.89 %	1.66 %	910.1	0.25 %	0.61 %	2349.3	0.07 %	0.44 %	7580.3
TAvg		2.28 %	3.33 %	113.3	0.44 %	0.86 %	996.9	0.21 %	0.44 %	1583.7	0.63 %	1.30 %	568.2	0.18 %	0.49 %	1450.7	0.06 %	0.34 %	4656.2

Table 10: Comparison of heuristics in the literature on homogenous instances with original pricing: Part 2.

[1] In the paper of Vidal et al. (2014), the running times are not given. These times were provided by him by email.

		RIP		TS+		TS		GA		R-TS			AVNS-Fast			AVNS-Slow		
		Bolduc et al. (2008)		Potvin and Naud (2011)		Potvin and Naud (2011)		Kratica et al. (2012)		Huijink et al. (2014)			this paper			this paper		
Instance	BKS	single	CPU(s)	best	CPU(s)	best	CPU(s)	best	CPU(s)	best	avg	CPU(s)	best	avg	CPU(s)	best	avg	CPU(s)
CE-H-01	1191.70	1192.72	26.0	1191.70	26.0	1191.70	25.7	1203.27	49.2	1191.70	1193.00	10.1	1191.70	1194.22	15.3	1191.70	1193.09	53.0
CE-H-02	**1789.41**	1798.26	72.0	1791.21	34.8	1795.51	33.7	1860.84	94.4	1795.91	1804.13	25.2	1796.13	1802.79	39.4	1789.41	1799.19	129.1
CE-H-03	**1916.81**	1934.85	105.0	1917.96	80.8	1926.33	79.0	1988.73	109.3	1922.82	1936.32	52.4	1916.81	1927.28	82.9	1917.46	1924.47	236.2
CE-H-04	**2468.63**	2493.93	251.0	2481.68	198.5	2481.64	195.6	2622.24	206.1	2484.79	2499.79	128.6	2477.69	2491.28	180.3	2468.63	2480.63	916.8
CE-H-05	**3123.07**	3195.66	490.0	3143.01	342.4	3143.92	295.9	3314.16	340.4	3152.15	3167.93	257.3	3135.83	3150.62	529.8	3123.07	3136.80	1682.0
CE-H-06	1204.48	1210.23	25.0	1206.82	25.1	1206.82	25.4	1210.75	51.5	1205.77	1209.52	11.9	1206.82	1211.36	16.0	1204.48	1207.75	59.9
CE-H-07	2025.98	2042.79	74.0	2031.85	32.0	2035.90	32.5	2108.23	90.4	2034.98	2041.41	24.6	2033.42	2039.31	44.2	2025.98	2032.50	119.8
CE-H-08	**1983.96**	2015.72	112.0	1986.51	84.5	1991.23	81.4	2057.75	119.2	1987.54	2004.60	53.1	1989.21	1995.29	92.2	1983.96	1987.58	326.7
CE-H-09	2425.06	2445.88	267.0	2447.58	193.0	2445.49	188.9	2601.96	233.7	2461.18	2471.66	129.5	2444.36	2453.97	264.5	2425.06	2446.58	1000.2
CE-H-10	**3250.15**	3304.69	482.0	3272.37	342.4	3271.70	309.5	3415.40	382.5	3271.86	3288.80	245.5	3267.88	3281.33	429.9	3254.66	3262.88	1937.1
CE-H-11	**2302.46**	2308.76	188.0	2336.51	133.9	2325.74	127.0	2381.52	139.5	2365.32	2418.32	88.7	2312.92	2371.09	144.2	2305.73	2313.60	527.6
CE-H-12	1908.05	1908.74	130.0	1915.05	60.4	1912.47	60.7	1954.80	109.7	1925.45	1937.72	50.9	1908.93	1934.57	65.3	1908.05	1919.17	192.8
CE-H-13	**2832.88**	2842.18	195.0	2868.13	136.5	2872.14	125.0	2883.67	143.9	2844.41	2925.09	77.2	2836.09	2863.02	104.0	2833.77	2847.97	404.7
CE-H-14	1907.74	1920.36	114.0	1907.75	67.2	1925.46	65.8	1988.79	111.6	1928.71	1948.96	51.4	1916.45	1929.76	72.6	1913.55	1925.31	198.9
Avg		0.83 %	180.8	0.47 %	125.5	0.58 %	117.6	3.83 %	155.8	0.74 %	1.57 %	86.2	0.31 %	0.99 %	148.6	0.05 %	0.47%	556.1
G-H-01	**14084.33**	14408.31	647.0					14812.40	661.9	14220.30	14315.62	332.8	14175.86	14197.77	643.2	14084.33	14123.48	3608.8
G-H-02	**18403.41**	18663.15	1254.0					19395.20	4757.0	18561.79	18737.67	699.9	18482.03	18533.04	2242.3	18403.41	18523.85	8184.2
G-H-03	**24949.88**	25561.55	2053.0					26523.43	1404.3	25315.53	25740.72	1119.7	25049.48	25209.63	2775.0	24949.88	25127.31	11445.1
G-H-04	**34147.04**	35495.66	2904.0					36261.53	23744.7	34654.94	35062.09	1679.5	34362.17	34476.52	4289.3	34147.04	34405.37	14821.2
G-H-05	**15423.65**	16138.50	512.0					16254.20	889.6	15647.09	15879.83	249.4	15423.65	15565.58	358.9	15423.65	15491.65	1619.8
G-H-06	**19695.40**	20329.04	1005.0					20717.86	5350.7	20030.05	20275.03	549.4	19742.57	19905.66	1161.2	19695.40	19868.59	4414.2
G-H-07	**23394.23**	24184.83	1608.0					24727.21	10955.3	23893.59	24162.40	886.0	23532.62	23638.61	2420.2	23394.23	23544.57	9853.6
G-H-08	**27111.30**	27710.66	2584.0	27334.84	18625.2	27521.28	3408.2	28605.47	23568.1	27446.57	27738.53	1452.9	27155.11	27305.97	4593.2	27111.30	27343.98	17610.4
G-H-09	**1325.09**	1346.03	814.0	1329.27	1829.3	1331.11	592.7	1386.03	3259.7	1346.36	1356.71	403.6	1330.30	1344.25	1060.9	1325.09	1330.04	4634.4
G-H-10	**1551.92**	1575.82	1332.0	1555.59	1564.3	1554.96	1087.1	1622.14	8750.4	1582.04	1591.64	724.0	1568.51	1575.60	1887.8	1551.92	1560.45	7680.1
G-H-11	**2188.14**	2218.91	2140.0	2195.83	3207.9	2191.23	1445.5	2266.04	14759.3	2216.85	2227.39	1110.8	2199.48	2216.19	3570.7	2188.14	2197.42	10844.3
G-H-12	2482.92	2510.07	2970.0	2482.92	4224.0	2535.00	2108.3	2580.32	32527.2	2539.58	2547.00	1773.3	2496.50	2519.29	8742.8	2491.08	2510.01	22103.2
G-H-13	**2216.80**	2253.45	733.0	2237.38	1801.3	2231.88	405.8	2330.81	1256.3	2260.02	2266.93	243.0	2229.22	2243.92	921.0	2216.80	2227.53	2819.4
G-H-14	**2653.58**	2711.81	1246.0	2684.70	1042.9	2685.51	630.3	2809.86	2687.7	2690.26	2726.82	343.0	2660.89	2675.40	1646.0	2653.58	2664.39	5595.0
G-H-15	**3111.08**	3156.93	1895.0	3127.33	2111.2	3123.60	976.6	3285.70	5963.1	3147.99	3168.74	853.5	3134.03	3148.68	2402.7	3111.08	3124.36	8458.0
G-H-16	**3608.27**	3649.09	2785.0	3621.85	6217.4	3853.21	1571.2	3780.43	10786.4	3672.97	3682.11	1334.9	3621.89	3649.42	4877.1	3608.27	3624.19	14078.1
G-H-17	**1689.30**	1705.48	762.0					1932.18	389.1	1705.92	1716.15	305.5	1696.72	1700.91	1068.0	1689.30	1695.24	2661.4
G-H-18	**2736.29**	2759.99	1299.0					3062.08	729.9	2757.40	2775.02	512.1	2738.17	2752.74	1741.4	2736.29	2740.59	3760.1
G-H-19	**3451.70**	3517.48	1892.0					3892.96	1004.0	3505.20	3534.22	841.5	3461.47	3486.10	2570.1	3451.70	3462.83	7257.6
G-H-20	**4321.06**	4413.82	2733.0					4865.32	1450.4	4399.10	4435.14	1167.4	4345.29	4364.39	3737.8	4321.06	4341.49	10005.3
Avg		2.08 %	1658.4	0.53 %	4513.7	1.50 %	1358.4	6.73 %	7744.7	1.49 %	2.35 %	829.1	0.45 %	1.03 %	2635.5	0.02 %	0.54%	8572.7
TAvg		1.56 %	1050.0	0.49 %	1842.7	0.94 %	603.1	5.53 %	4619.9	1.18 %	2.03 %	523.2	0.39 %	1.02 %	1611.5	0.03 %	0.51%	5271.7

Table 11: Comparison of heuristics in the literature on heterogenous instances with original pricing.

		R-TS			AVNS-Fast			AVNS-Slow		
		Huijink et al. (2014)			this paper			this paper		
Instance	BKS	best	avg	CPU(s)	best	avg	CPU(s)	best	avg	CPU(s)
CE-01	940.66	940.66	940.66	8.2	940.66	940.66	18.4	940.66	940.66	53.4
CE-02	1605.51	1605.51	1608.00	15.1	1605.51	1607.48	26.1	1605.51	1607.05	122.3
CE-03	1706.18	1706.18	1723.93	28.4	1706.18	1706.78	58.8	1706.18	1706.66	243.1
CE-04	2237.96	2247.35	2259.89	97.6	2245.37	2250.13	189.3	2237.96	2243.47	515.5
CE-05	2770.34	2795.24	2818.86	149.0	2777.37	2791.48	328.7	2770.34	2778.85	1481.1
CE-06	985.66	985.66	985.66	7.8	985.66	985.66	18.6	985.66	985.66	63.0
CE-07	1721.33	1721.33	1722.70	14.3	1721.33	1721.33	26.5	1721.33	1721.33	129.7
CE-08	1804.49	1804.68	1808.73	25.4	1804.49	1804.49	61.7	1804.49	1804.49	237.1
CE-09	2250.46	2259.03	2272.57	91.4	2254.69	2261.07	166.3	2250.68	2255.55	807.6
CE-10	3024.15	3037.10	3051.76	152.8	3025.90	3034.87	426.9	3024.36	3031.39	1361.9
CE-11	2172.90	2172.90	2187.15	45.6	2172.90	2172.90	64.9	2172.90	2172.90	439.7
CE-12	1800.85	1800.85	1817.46	36.9	1800.85	1802.52	57.2	1800.85	1800.85	164.4
CE-13	2626.79	2626.79	2641.49	49.5	2626.79	2626.79	58.5	2626.79	2626.79	385.2
CE-14	1922.85	1932.46	1933.59	34.7	1922.85	1925.38	47.2	1922.85	1927.92	203.9
Avg		0.19 %	0.63 %	54.0	0.06 %	0.18 %	110.6	0.00 %	0.10 %	443.4
G-01	12436.61	12515.67	12575.64	380.2	12443.83	12473.49	584.6	12436.61	12470.22	1805.9
G-02	17776.73	18085.95	18177.09	723.1	17785.73	17888.16	1449.8	17798.71	17851.93	6159.2
G-03	23059.15	23542.31	23791.36	1380.6	23059.15	23161.63	3802.2	23093.44	23187.79	10620.9
G-04	30502.95	31401.51	31643.91	2127.9	30502.95	30743.75	6024.1	30582.02	30701.44	15880.1
G-05	13504.63	13620.64	13808.72	274.0	13549.86	13565.68	442.0	13510.42	13539.15	1203.6
G-06	18804.73	19131.60	19316.61	589.7	18810.01	18943.16	1356.5	18804.73	18904.76	3196.8
G-07	21676.11	22198.97	22338.11	1047.8	21676.11	21835.66	3004.9	21701.00	21759.72	8149.5
G-08	26262.70	26795.63	26956.80	1639.6	26262.70	26437.72	3838.7	26270.65	26369.04	11898.5
G-09	1208.17	1210.49	1215.13	357.1	1210.15	1213.28	1017.3	1208.17	1210.48	3028.9
G-10	1459.31	1467.58	1472.40	619.0	1461.32	1467.08	2686.0	1459.31	1463.96	5932.9
G-11	1989.39	2003.58	2008.89	1011.5	1990.65	2001.80	3357.5	1992.44	1999.34	10556.3
G-12	2308.00	2331.91	2337.43	1594.8	2313.66	2326.35	4292.4	2308.00	2321.32	16791.0
G-13	1864.50	1871.21	1876.80	295.2	1867.66	1873.54	660.7	1864.50	1869.14	2382.9
G-14	2273.69	2278.74	2289.17	538.8	2275.42	2282.32	1190.6	2273.69	2279.73	3939.8
G-15	2727.13	2742.70	2747.59	803.1	2728.51	2736.12	2371.4	2727.13	2733.24	6804.9
G-16	3194.65	3214.05	3220.45	1425.5	3201.93	3209.86	3811.3	3194.65	3203.14	11057.4
G-17	1506.42	1514.64	1520.32	244.5	1508.83	1510.51	447.1	1506.42	1508.23	1794.6
G-18	2420.46	2430.18	2436.72	520.7	2424.70	2426.59	840.5	2420.46	2422.93	3122.4
G-19	3099.49	3118.98	3131.01	756.3	3106.97	3112.98	1618.6	3099.49	3105.88	5261.1
G-20	3886.86	3915.36	3923.88	986.0	3888.46	3898.38	2551.2	3886.86	3892.43	8428.8
Avg		1.05 %	1.56 %	865.8	0.11 %	0.50 %	2267.4	0.04 %	0.33 %	6900.8
Tavg		0.70 %	1.19 %	532.9	0.09 %	0.37 %	1382.0	0.03 %	0.24 %	4253.3

		R-TS			AVNS-Fast			AVNS-Slow		
		Huijink et al. (2014)			this paper			this paper		
Instance	BKS	best	avg	CPU(s)	best	avg	CPU(s)	best	avg	CPU(s)
CE-H-01	977.38	977.38	983.18	8.4	977.38	977.38	16.9	977.38	977.38	52.3
CE-H-02	1583.65	1585.32	1592.20	18.9	1585.32	1585.47	33.4	1585.32	1585.33	147.1
CE-H-03	1688.51	1708.63	1724.24	30.0	1688.51	1692.05	48.2	1688.51	1691.50	234.8
CE-H-04	2205.69	2214.48	2250.90	90.0	2207.15	2213.60	199.3	2205.69	2213.08	729.9
CE-H-05	2791.55	2822.16	2854.05	161.0	2804.41	2815.80	310.0	2791.55	2801.69	1662.1
CE-H-06	977.38	977.38	986.10	7.6	977.38	977.38	12.6	977.38	977.38	58.3
CE-H-07	1704.52	1713.37	1732.40	15.8	1704.52	1705.44	27.2	1704.52	1704.52	129.0
CE-H-08	1789.60	1796.14	1814.53	31.9	1790.59	1795.16	55.5	1789.60	1792.03	312.4
CE-H-09	2229.19	2260.34	2282.47	94.3	2236.01	2244.82	166.6	2232.48	2238.13	741.2
CE-H-10	2958.74	2992.47	3020.20	164.1	2968.81	2977.85	359.3	2958.74	2966.11	1644.8
CE-H-11	2144.23	2163.30	2232.65	52.6	2144.23	2164.79	81.4	2146.69	2167.80	312.0
CE-H-12	1759.78	1784.74	1804.19	39.1	1759.78	1768.62	53.5	1759.78	1765.97	179.7
CE-H-13	2611.25	2625.99	2666.18	52.0	2611.25	2617.35	71.3	2611.25	2616.82	310.1
CE-H-14	1742.97	1749.66	1766.54	34.2	1742.97	1744.69	43.6	1742.97	1743.13	201.8
Avg		0.68 %	1.86 %	57.1	0.10 %	0.36 %	105.6	0.03 %	0.25 %	479.7
G-H-01	12388.53	12468.29	12562.88	327.7	12388.53	12458.77	812.6	12388.67	12450.85	2214.1
G-H-02	17647.94	17911.18	18015.02	683.1	17680.53	17755.04	1846.4	17647.94	17712.64	6188.3
G-H-03	23185.22	23705.36	23837.41	1153.8	23196.85	23365.56	3637.1	23217.38	23303.71	12854.1
G-H-04	30562.87	31515.65	31754.33	1719.2	30562.87	30745.98	5750.9	30595.08	30727.85	15815.3
G-H-05	13628.09	13989.77	14088.29	209.6	13737.34	13757.65	387.7	13628.09	13708.55	1111.3
G-H-06	18657.87	19139.17	19309.62	492.1	18698.61	18820.31	1153.7	18657.87	18732.83	4321.8
G-H-07	21667.02	22307.68	22410.60	878.2	21677.33	21818.77	2937.1	21667.02	21755.58	8852.1
G-H-08	25605.72	26220.00	26373.02	1425.8	25605.72	25825.78	4948.0	25701.52	25846.74	15225.8
G-H-09	1189.89	1196.27	1219.74	317.2	1191.10	1199.68	1206.1	1189.89	1196.42	3085.1
G-H-10	1419.46	1430.80	1444.11	588.2	1419.67	1426.49	2272.4	1419.46	1425.22	5763.5
G-H-11	1994.00	2003.06	2028.53	984.7	1998.34	2016.69	3670.1	1994.00	2004.07	11671.8
G-H-12	2294.19	2319.86	2349.60	1402.3	2296.31	2316.75	7551.0	2294.19	2304.20	16820.4
G-H-13	1822.11	1836.60	1850.56	261.0	1824.70	1831.04	770.5	1822.11	1827.18	1929.4
G-H-14	2216.72	2234.72	2253.12	468.5	2217.67	2225.52	1826.2	2216.72	2224.22	3650.2
G-H-15	2665.05	2690.17	2710.19	760.4	2665.05	2675.88	3163.2	2668.01	2675.72	6414.7
G-H-16	3133.58	3158.71	3183.54	1237.2	3138.99	3144.90	4869.7	3133.58	3144.02	10913.1
G-H-17	1484.19	1496.98	1509.74	234.7	1490.03	1494.31	508.3	1484.19	1488.67	2381.9
G-H-18	2389.68	2416.87	2443.22	419.7	2399.25	2403.82	1343.3	2389.68	2393.91	4120.3
G-H-19	3039.23	3104.18	3118.94	670.9	3052.30	3063.66	1937.3	3039.23	3049.80	6645.9
G-H-20	3832.45	3901.84	3930.85	981.8	3850.45	3865.79	3070.5	3832.45	3847.81	8697.7
Avg		1.51 %	2.38 %	760.8	0.19 %	0.70 %	2683.1	0.04 %	0.44 %	7433.8
Tavg		1.19 %	2.20 %	472.5	0.15 %	0.57 %	1624.4	0.03 %	0.37 %	4582.9

Table 12: Comparison of our heuristics on the instances with half-original pricing.

		R-TS Huijink et al. (2014)			AVNS-Fast this paper			AVNS-Slow this paper		
Instance	BKS	best	avg	CPU(s)	best	avg	CPU(s)	best	avg	CPU(s)
CE-01	1082.37	1082.37	1084.47	6.7	1082.37	1083.71	11.1	1082.37	1082.37	68.7
CE-02	1764.02	1769.18	1772.25	14.3	1764.02	1768.74	24.7	1764.02	1765.48	143.0
CE-03	1779.45	1787.64	1796.86	30.2	1783.00	1785.13	64.7	1782.66	1783.89	242.2
CE-04	2290.51	2305.44	2319.44	80.9	2293.07	2301.22	180.7	2291.79	2294.44	736.7
CE-05	2780.37	2788.17	2799.96	146.5	2784.57	2792.04	235.1	2780.37	2784.69	1232.4
CE-06	1202.26	1202.26	1208.13	7.0	1202.26	1202.62	12.6	1202.26	1202.26	36.7
CE-07	1983.37	1987.44	1997.00	17.4	1983.37	1987.70	37.7	1983.37	1983.52	117.4
CE-08	1948.95	1950.02	1966.43	32.2	1948.95	1952.61	78.0	1948.95	1949.99	185.9
CE-09	2310.51	2330.64	2341.48	67.0	2318.01	2325.42	183.7	2310.51	2318.37	606.9
CE-10	3150.98	3157.35	3171.65	182.5	3158.20	3165.19	407.3	3152.73	3157.29	1105.9
CE-11	2203.91	2204.53	2211.57	45.8	2204.53	2205.18	95.2	2205.16	2205.38	363.0
CE-12	1894.10	1894.10	1902.76	36.7	1894.10	1896.49	52.8	1894.10	1894.10	172.5
CE-13	2857.77	2857.91	2865.42	79.3	2857.77	2861.62	119.8	2857.77	2859.08	333.4
CE-14	2083.85	2091.04	2117.77	21.7	2083.85	2084.82	75.7	2083.85	2084.95	175.8
Avg		0.24 %	0.74 %	54.9	0.07 %	0.25 %	112.8	0.02 %	0.10 %	394.3
G-01	12160.83	12167.03	12203.75	274.3	12171.99	12199.00	578.0	12160.83	12172.32	2209.0
G-02	17255.76	17305.88	17340.51	536.4	17277.95	17310.63	1358.9	17272.27	17281.46	5784.7
G-03	22469.01	22538.27	22578.90	975.3	22501.25	22524.09	2197.0	22469.01	22483.14	9927.6
G-04	29646.47	29724.59	29782.63	1496.0	29688.87	29719.40	4954.5	29646.47	29695.83	14295.0
G-05	13660.87	13775.64	13809.26	237.2	13668.69	13678.37	524.7	13660.87	13666.51	2032.7
G-06	18939.58	18988.28	19006.62	425.4	18941.45	18958.08	1179.8	18939.58	18959.35	4827.7
G-07	21258.96	21274.59	21344.67	806.7	21266.90	21309.52	2225.0	21258.96	21279.35	6528.5
G-08	25245.47	25342.60	25379.77	1181.7	25279.78	25314.52	3161.7	25245.47	25305.86	10793.8
G-09	1425.30	1427.05	1440.65	354.3	1427.97	1432.77	945.2	1425.30	1429.17	1980.7
G-10	1680.21	1690.15	1696.69	592.4	1683.72	1687.41	2070.6	1680.21	1684.13	4355.4
G-11	2337.93	2364.69	2378.77	998.9	2350.42	2365.70	3029.8	2337.93	2348.90	8794.1
G-12	2666.87	2696.75	2715.36	1450.7	2679.29	2691.56	5267.5	2666.87	2680.82	11648.1
G-13	2450.61	2456.34	2474.08	320.7	2450.61	2460.97	784.6	2450.88	2456.49	1793.6
G-14	2902.98	2910.44	2933.74	541.2	2907.96	2919.13	1149.7	2902.98	2908.67	3473.8
G-15	3377.13	3408.68	3422.22	856.7	3381.12	3401.22	2007.1	3377.13	3392.05	5030.0
G-16	3864.52	3885.17	3898.93	1213.7	3873.89	3887.29	3777.5	3864.52	3876.37	7499.2
G-17	1504.53	1510.87	1516.40	270.0	1504.59	1508.14	537.3	1504.53	1506.09	1471.8
G-18	2584.71	2601.64	2607.96	455.7	2588.28	2593.54	1440.0	2584.71	2588.30	3604.8
G-19	3187.87	3205.69	3207.08	664.4	3192.01	3199.29	1790.0	3187.87	3191.99	4608.1
G-20	3816.89	3839.71	3847.94	972.3	3826.09	3834.80	2723.2	3816.89	3823.27	6486.3
Avg		0.48 %	0.86 %	731.2	0.16 %	0.43 %	2085.1	0.01 %	0.21 %	5857.2
Tavg		0.39 %	0.83 %	454.1	0.13 %	0.36 %	1276.0	0.01 %	0.17 %	3617.4

		R-TS Huijink et al. (2014)			AVNS-Fast this paper			AVNS-Slow this paper		
Instance	BKS	best	avg	CPU(s)	best	avg	CPU(s)	best	avg	CPU(s)
CE-H-01	1189.28	1190.90	1194.70	7.5	1189.28	1190.69	12.2	1189.28	1189.96	48.7
CE-H-02	1676.73	1676.73	1685.66	16.8	1676.73	1680.68	25.8	1676.73	1677.66	116.2
CE-H-03	1714.44	1732.17	1739.06	38.6	1716.18	1719.61	50.7	1714.44	1719.19	224.6
CE-H-04	2216.66	2220.75	2237.37	90.8	2221.82	2226.70	154.3	2216.66	2222.46	560.8
CE-H-05	2720.67	2738.90	2750.17	159.8	2729.34	2738.32	263.9	2721.72	2730.18	1208.1
CE-H-06	1167.81	1173.22	1174.83	7.0	1168.29	1169.50	13.0	1167.81	1168.06	53.7
CE-H-07	1945.28	1956.60	1963.40	18.0	1945.28	1954.24	29.2	1945.28	1955.11	134.3
CE-H-08	1881.24	1890.61	1900.02	31.1	1883.61	1887.29	62.7	1881.24	1885.15	277.6
CE-H-09	2248.22	2266.16	2281.09	86.8	2261.08	2266.24	150.7	2248.22	2254.21	719.5
CE-H-10	3043.12	3053.87	3070.88	154.1	3045.31	3058.24	307.0	3044.75	3050.71	1440.8
CE-H-11	2087.08	2087.67	2098.38	52.2	2087.43	2088.90	110.9	2087.08	2090.06	443.5
CE-H-12	1810.05	1813.47	1822.05	33.0	1810.56	1815.12	59.0	1810.05	1813.13	279.5
CE-H-13	2785.92	2791.48	2826.10	47.2	2791.48	2802.87	80.6	2791.26	2800.95	464.3
CE-H-14	1764.69	1767.39	1778.25	32.2	1764.69	1766.14	72.7	1764.69	1765.76	220.2
Avg		0.38 %	0.91 %	55.4	0.12 %	0.36 %	99.5	0.02 %	0.23 %	442.3
G-H-01	11600.91	11617.17	11687.57	284.3	11608.17	11625.17	731.1	11600.91	11616.10	2280.8
G-H-02	16296.86	16304.96	16444.74	624.9	16296.86	16339.31	1258.0	16297.17	16350.30	5116.4
G-H-03	21192.11	21316.83	21370.25	978.7	21192.11	21226.30	2578.3	21198.02	21219.08	8866.8
G-H-04	27963.45	28043.39	28097.85	1296.1	27963.45	27983.59	3463.7	27963.45	27993.34	11015.6
G-H-05	12978.64	12978.64	13002.80	164.9	12978.64	12978.64	268.1	12978.64	12978.64	1380.4
G-H-06	18744.81	18785.11	18844.10	456.9	18744.81	18775.05	1099.8	18744.81	18764.69	4261.1
G-H-07	20014.52	20058.75	20182.30	741.7	20014.52	20036.97	1886.5	20014.52	20025.81	6477.2
G-H-08	23880.67	23975.58	24050.09	1287.5	23888.10	23953.45	2982.3	23880.67	23938.39	9554.3
G-H-09	1397.97	1417.24	1426.17	309.0	1399.27	1410.64	1139.7	1397.97	1401.84	2576.1
G-H-10	1586.71	1597.32	1601.70	407.5	1588.24	1595.56	1421.8	1586.71	1591.24	3268.0
G-H-11	2324.40	2360.26	2373.82	835.5	2340.72	2350.73	3577.2	2324.40	2340.80	7337.6
G-H-12	2613.29	2653.53	2666.66	882.3	2616.15	2630.18	5505.1	2613.29	2620.57	10575.8
G-H-13	2394.41	2420.39	2426.19	305.1	2395.42	2410.26	685.9	2394.41	2399.65	1986.0
G-H-14	2837.66	2881.24	2892.25	471.7	2847.41	2868.13	1533.7	2837.66	2851.31	3676.1
G-H-15	3285.27	3311.16	3319.57	659.9	3285.27	3297.73	2240.3	3288.63	3295.72	4561.0
G-H-16	3750.00	3758.22	3769.44	933.7	3750.00	3756.17	3406.9	3750.03	3755.35	6594.7
G-H-17	1425.48	1434.22	1440.64	259.1	1427.61	1429.10	612.2	1425.48	1427.18	1631.5
G-H-18	2519.91	2541.13	2547.82	474.2	2523.32	2527.72	967.6	2519.91	2522.79	3641.2
G-H-19	3055.15	3070.46	3087.22	684.5	3060.89	3069.05	1806.5	3055.15	3061.84	5478.8
G-H-20	3669.77	3690.25	3712.09	931.2	3684.28	3692.04	2616.3	3669.77	3680.01	8217.1
Avg		0.66 %	1.08 %	649.4	0.12 %	0.42 %	1989.0	0.01 %	0.23 %	5424.8
Tavg		0.55 %	1.02 %	406.2	0.12 %	0.40 %	1213.5	0.01 %	0.23 %	3384.8

Table 13: Comparison of our heuristics on the instances with new pricing.

Move	G-01	G-02	G-03	G-04	G-05	G-06	G-07	G-08	G-09	G-10	G-11	G-12	G-13	G-14	G-15	G-16	G-17	G-18	G-19	G-20	avg
1	4.7	8.5	8.3	9.5	7.3	8.9	5.4	10.2	4.7	7.8	4.0	7.8	8.4	10.5	10.0	10.5	6.2	12.9	11.4	11.0	8.4
2	5.2	7.0	7.3	8.4	7.7	5.6	6.0	8.0	5.1	7.5	6.2	9.3	7.8	7.4	7.7	8.7	5.0	13.6	9.6	10.4	7.7
3	4.2	5.0	7.0	4.2	3.3	4.3	5.5	6.7	2.5	3.7	2.8	4.4	4.0	4.2	3.4	4.8	2.5	6.5	4.7	5.2	4.4
4	4.5	3.8	4.4	3.1	0.8	0.8	4.8	3.5	1.2	1.9	1.9	2.4	3.2	1.8	2.2	2.4	1.0	2.7	2.5	4.7	2.7
5	5.3	8.5	8.1	8.2	5.9	9.5	7.3	10.0	10.2	12.4	15.9	18.7	3.9	7.5	9.9	10.9	4.1	5.9	6.2	5.7	8.7
6	4.3	8.7	7.9	8.7	4.4	7.9	6.8	9.0	6.7	9.9	10.6	14.8	4.6	7.6	7.1	11.9	4.4	5.9	5.4	5.7	7.6
7	3.1	4.1	6.5	3.9	1.3	2.9	4.4	3.9	3.3	4.8	4.5	5.3	2.3	3.5	4.1	4.5	1.3	2.4	1.7	2.3	3.5
8	1.1	0.8	1.1	0.6	1.2	0.9	0.5	1.4	0.7	2.6	1.8	3.0	0.5	0.6	1.7	5.5	0.9	1.6	2.0	1.7	1.5
9	0.2	0.3	0.2	0.1	0.2	0.2	0.4	0.2	0.4	2.0	0.6	1.6	0.0	0.6	0.6	1.2	1.0	0.6	1.2	0.8	0.6
10	0.7	1.4	0.2	0.3	1.6	0.9	0.6	0.7	0.9	3.1	1.9	3.7	0.2	0.8	1.6	6.9	2.0	1.9	1.7	1.8	1.6
11	4.0	4.6	4.3	3.8	6.0	5.9	3.4	4.4	5.1	6.3	7.6	7.0	11.7	12.8	10.0	9.4	5.4	4.5	4.4	5.0	6.3
12	1.6	2.0	2.1	2.3	5.1	4.8	3.8	2.4	3.8	2.6	3.9	3.1	5.8	11.6	7.5	6.1	1.1	0.6	0.6	1.2	3.6
13	5.3	10.5	8.3	10.4	5.2	11.3	8.1	10.0	10.9	8.8	16.8	9.4	3.5	9.8	9.2	7.8	5.4	8.8	7.7	6.2	8.7
14	5.3	7.2	8.0	7.3	2.0	6.4	7.6	9.5	8.6	4.9	10.7	6.7	2.3	5.0	6.1	4.8	2.2	6.0	4.5	3.6	5.9
15	2.2	2.6	1.6	2.9	5.8	3.8	2.2	1.9	2.3	2.6	3.4	3.5	5.1	2.6	5.9	5.3	4.5	5.4	3.5	3.0	3.5
16	1.0	1.7	1.2	1.8	5.6	3.9	1.6	1.1	1.9	0.8	1.7	1.2	3.5	1.6	1.7	2.3	3.6	1.3	1.2	1.5	2.0
17	2.4	4.0	5.3	4.1	3.6	6.1	3.9	6.5	12.5	17.5	19.2	22.8	14.0	17.9	16.8	15.6	7.8	9.5	9.8	10.0	10.5

Table 14: The average number of times each move found an improvement with new pricing.

Move	G-01	G-02	G-03	G-04	G-05	G-06	G-07	G-08	G-09	G-10	G-11	G-12	G-13	G-14	G-15	G-16	G-17	G-18	G-19	G-20	avg
1	7.7	4.2	6.3	5.7	5.0	3.5	6.3	3.7	1.7	0.8	1.8	1.2	3.0	1.4	0.9	2.3	0.7	0.8	0.9	1.1	2.7
2	7.7	6.9	4.0	5.3	4.7	2.4	9.0	5.5	2.3	1.3	1.8	1.2	2.8	1.7	1.6	1.6	0.6	0.9	1.0	1.5	3.0
3	9.0	7.6	11.7	7.2	7.6	3.0	8.4	11.5	2.1	2.3	5.3	1.9	2.9	4.8	2.9	3.2	0.9	0.9	1.1	2.1	5.3
4	7.8	7.6	8.3	7.6	10.1	2.7	4.4	11.6	2.6	2.5	2.9	4.4	3.5	3.1	6.5	3.9	1.1	3.1	1.6	2.1	5.3
5	5.2	4.4	6.0	5.8	4.3	1.9	2.5	4.4	1.1	0.8	1.2	1.2	6.3	2.5	1.7	2.2	0.9	2.0	1.4	1.5	2.6
6	4.7	6.0	6.4	5.2	7.4	2.6	4.0	4.5	1.2	1.2	1.4	1.1	3.2	2.4	1.6	1.6	0.9	0.9	1.0	2.5	2.8
7	8.8	4.1	9.2	7.5	6.2	6.7	7.3	12.7	2.3	1.6	3.7	1.7	6.0	6.2	2.4	2.3	1.2	2.0	2.9	2.1	5.1
8	24.3	21.3	40.2	15.0	46.8	16.6	6.4	49.5	11.5	8.1	24.7	8.0	53.7	16.4	16.3	5.3	3.5	4.3	6.0	4.1	15.3
9	2.6	12.8	18.0	40.1	2.3	8.8	14.5	74.1	1.1	2.7	4.8	3.8	.	18.3	11.1	4.2	1.5	5.1	2.6	1.9	6.6
10	47.0	42.4	171.5	33.6	38.0	18.6	11.8	44.9	18.9	8.9	30.0	13.0	56.1	32.8	20.4	6.0	1.3	3.4	4.8	4.7	16.4
11	7.8	9.8	7.5	14.4	6.5	2.4	8.4	7.4	2.0	1.7	2.5	2.6	4.8	2.9	3.7	2.5	0.9	1.3	2.4	1.9	4.1
12	10.5	8.8	10.1	9.4	6.9	5.6	11.6	12.2	2.8	1.9	6.5	4.3	9.8	5.5	6.2	3.9	1.3	2.5	4.6	2.1	6.5
13	4.2	6.1	9.3	7.4	4.0	2.5	6.0	4.2	1.4	0.9	1.4	0.8	5.0	3.5	1.6	2.5	0.5	0.8	1.0	1.2	3.1
14	6.0	3.3	3.2	9.5	3.9	3.1	5.5	6.3	1.9	1.6	1.2	1.5	8.0	4.7	3.4	2.5	0.6	0.9	1.5	1.1	3.5
15	3.5	8.7	23.9	14.2	8.2	3.9	9.8	23.3	3.1	3.6	4.8	2.9	4.1	7.8	4.4	4.1	1.0	1.2	1.1	2.6	5.6
16	8.4	8.4	19.4	9.8	6.8	9.3	33.0	34.1	4.9	2.4	9.0	4.0	5.3	14.0	7.3	7.1	0.9	3.6	3.8	8.5	8.8
17	13.1	3.2	5.7	12.8	5.9	2.1	9.7	5.4	0.7	0.8	0.8	0.8	2.3	1.4	1.1	1.5	0.5	0.8	0.9	1.1	2.0

Table 15: The average value of each improvement with new pricing.